UNIVERSIDAD
COMPLUTENSE
MADRID

Máster en Ingeniería Informática

Herramienta lingüística de manipulación de oraciones y escritura de gráficos basada en el procesamiento del lenguaje natural

---

Linguistic tool for sentence processing and graph writing based on natural language processing

Realizado por:

Carlos Gavidia Ortiz

Directores:

Antonio Sarasa Cabezuelo
Elena del Olmo Suárez
Ana María Fernández-Pampillón Cesteros

Facultad de Informática
Universidad Complutense de Madrid
Convocatoria: Julio 2020
Calificación obtenida: 8

II. Agradecimientos

No puedo comenzar la memoria de este proyecto sin antes agradecer la ayuda y la confianza que me han aportado mis directores, Antonio Sarasa Cabezuelo, Elena del Olmo Suárez y Ana María Fernández-Pampillón Cesteros. Muchas gracias por los consejos y dudas resueltas, especialmente en aspectos sobre reglas, patrones, corpus lingüísticos y más conceptos que aparecen en el proyecto y que fueron esenciales para el desarrollo. Sin ellos, probablemente este proyecto no podría haber sido desarrollado.

Gracias a mi familia, amigos y mi novia por el apoyo, ánimos recibidos y aguantarme durante el desarrollo del proyecto y sobretodo durante estos 2 años. Sin la energía que me transmitían sobretodo en los peores momentos hubiera sido imposible terminar este proyecto.

Finalmente, quiero agradecer a mis profesores que me han dado clases durante el curso y a mis amigos de la facultad que han aportado su granito de arena.

III. Resumen

El proceso para analizar un texto, es una tarea de interés entre los lingüistas, ya que con ello se pueden realizar diversas funciones como la posibilidad de encontrar patrones de palabras que se repitan o incluso llegar a simplificar el texto eliminado las frases que no aporten una información relevante. Para conseguir estos objetivos es necesario el análisis de cada palabra del texto y estudiar como dependen las palabras entre ellas. Es por tanto una tarea muy costosa y que puede conllevar bastante tiempo de realizar de una forma manual.

El objetivo principal del presente proyecto es tratar de ayudar a los usuarios en estas problemáticas automatizando dos tareas. Por un lado, solucionar el problema del análisis textual, ya que se hará de una forma automática, se estudiarán tanto las palabras que lo componen, como las dependencias que existan entre ellas. Y por otro, se facilitará al usuario la búsqueda de patrones de palabras, y la creación de reglas para la simplificación del texto, en base a los criterios que determine el usuario.

Para cubrir el objetivo anterior, se ha implementado una aplicación web que permite a un usuario analizar textos y visualizar los resultados de una manera gráfica mostrando las oraciones como un grafo arbóreo de las palabras que lo componen. Además mediante la integración de la herramienta Grew, el usuario puede manipular el árbol de análisis generado, utilizando reglas de transformación o bien buscar patrones lingüísticos sobre el texto analizado.

Palabras clave

- Análisis textual
- Simplificar
- Reglas y patrones lingüísticos
- Aplicación web
- Grafo arbóreo
- Grew

IV. Abstract

The process to analyze a text is a task of interest among linguists, since various functions can be performed with this, such as the possibility of finding repeating word patterns or even simplifying the text by eliminating phrases that do not provide a relevant information. To achieve these objectives, it is necessary to analyze each word of the text and study how the words depend on each other. It is therefore a very expensive task and can take a long time to perform manually.

The main objective of this project is to try to help users in these problems by automating two tasks. On the one hand, solving the problem of textual analysis, since it will be done automatically, will study both the words that compose it, and the dependencies that exist between them. And on the other hand, the user will find the search for word patterns, and the creation of rules to simplify the text, based on the criteria determined by the user.

To cover the previous objective, a web application has been implemented that allows a user to analyze texts and visualize the results in a graphic way, showing the sentences as a tree graph of the words that compose it. Furthermore, by integrating the Grew tool, the user can manipulate the generated analysis tree, using transformation rules or search for linguistic patterns on the analyzed text.

KeyWords

- Textual analysis
- Simplify
- Rules and linguistic patterns
- Web application
- Tree graph
- Grew

Índice

Capítulo 1: Introducción

En este primer capitulo se presentará la motivación del proyecto, los objetivos planteados y el plan de trabajo seguido para organizar la investigación e implementación del proyecto.

1.1. Motivación

Una tarea esencial de la filología, ha sido y sigue siendo el estudio de las palabras, su origen, su morfología, a que categoría gramatical pertenecen, si son adjetivos, sustantivos, si son verbos, que preposiciones existen... Una vez se tenían los conocimientos sobre todos los tipos de palabras que existen, el siguiente paso era juntar diferentes palabras y formar frases con coherencia y cohesión, y proceder al estudio de como dependen las palabras unas de otras. Consiguiendo por ejemplo encontrar el sujeto, el predicado, el complemento directo, indirecto... Tradicionalmente esta tarea se ha denominado como realizar un análisis sintáctico de las oraciones. El estudio de observar como dependen las palabras unas de otras en las oraciones, es un proceso necesario para encontrar el significado clave de las oraciones y centrarse en el mensaje fundamental que quiere transmitir un texto.

Este proceso puede llegar a ser muy sencillo, en caso de tener que analizar solo una frase de unas cuantas palabras, o muy largo y costoso si se deben analizar varios textos que los compongan muchas palabras. Esta dificultad se debe, a que por cada oración hay que analizar cada una de las palabras y estudiar cómo dependen unas de otras. Una vez haya terminado este proceso de análisis, se podrá realizar cualquier funcionalidad que queramos, como simplificar el texto eliminado proposiciones no necesarias, buscar cuantas veces se repiten diferentes palabras o averiguar la temática del texto, incluso se puede llegar a conocer que pensamientos ha podido llegar a tener el autor para escribir el texto.

Este proyecto se centra en el análisis de cualquier texto, con el fin ayudar en ese proceso tan costoso. Automatizando tareas como el análisis y dependencias de las palabras que componen el texto. Gracias a esto el usuario podrá invertir el tiempo directamente en realizar todo lo que tenia pensado hacer después de haberlo analizado.

1.2. Objetivos

El objetivo en este proyecto, ha sido desarrollar una aplicación web para analizar cualquier corpus lingüístico que contenga un conjunto de textos, y ofrecer al usuario la posibilidad de buscar patrones, realizar reglas lingüísticas y representar de una forma gráfica los resultados. Este objetivo principal se puede precisar en varios subobjetivos:

- Diseñar una herramienta intuitiva y sencilla orientada tanto a usuarios con conocimientos lingüísticos, como a usuarios que carezcan de esos conocimientos, en dos tipos de interfaces diferentes.
- Permitir a cualquier usuario cargar cualquier tipo de corpus, de diferentes idiomas, y de diferentes formatos.
- Mostrar al usuario los resultados hallados durante el análisis, y los tipos de oraciones cargadas.
- Almacenar la información morfológica de todas las palabras de los textos cargados.
- Habilitar una sección al usuario con conocimientos lingüísticos, para poder escribir patrones y así realizar una búsqueda en el corpus analizado, e informar al usuario de los resultados obtenidos tras ese proceso.

- Conceder al usuario un editor para que pueda escribir reglas lingüísticas, aplicarlas a las oraciones del corpus cargado y así poder reducir las palabras que componen una oración.
- Visualizar las oraciones como un grafo en forma de árbol de dependencias, donde las palabras de la oración serán los nodos, y observar toda la información morfológica de cada palabra.
- Guardar y deshacer las operaciones realizadas de cada oración.
- Descargar todo el proceso de análisis del corpus realizado en cualquier momento.
- Ofrecer un formulario sencillo para realizar la creación de algunas reglas para la simplificación de oraciones.

1.3. Plan de trabajo

Para el desarrollo del proyecto se ha realizado el siguiente plan de trabajo:

1. Investigación sobre herramientas similares. En esta primer fase se hizo una búsqueda de aplicaciones que tuvieran una funcionalidad similar al del proyecto, para tomar como referencia diferentes aspectos de diseño y de funcionalidad.
2. Especificación de requisitos. En esta fase se determinaron todas las funcionalidades que se iban a implementar en el proyecto.
3. Diseño de la aplicación. En esta fase se diseñaron los bocetos de cómo se pensó, que se estructurarían los elementos y la interfaz al usuario.
4. Desarrollo de implementación y diseño. En esta fase se desarrolló la herramienta con todas las funcionalidades propuestas y una interfaz intuitiva y visual.
5. Pruebas y evaluación final. Durante esta fase se realizaron pruebas funcionales para detectar errores y problemas, y poderlos solucionarlos.

En la Figura 1 se puede apreciar el cronograma de las tareas realizadas para el desarrollo del proyecto y una estimación de las fechas.

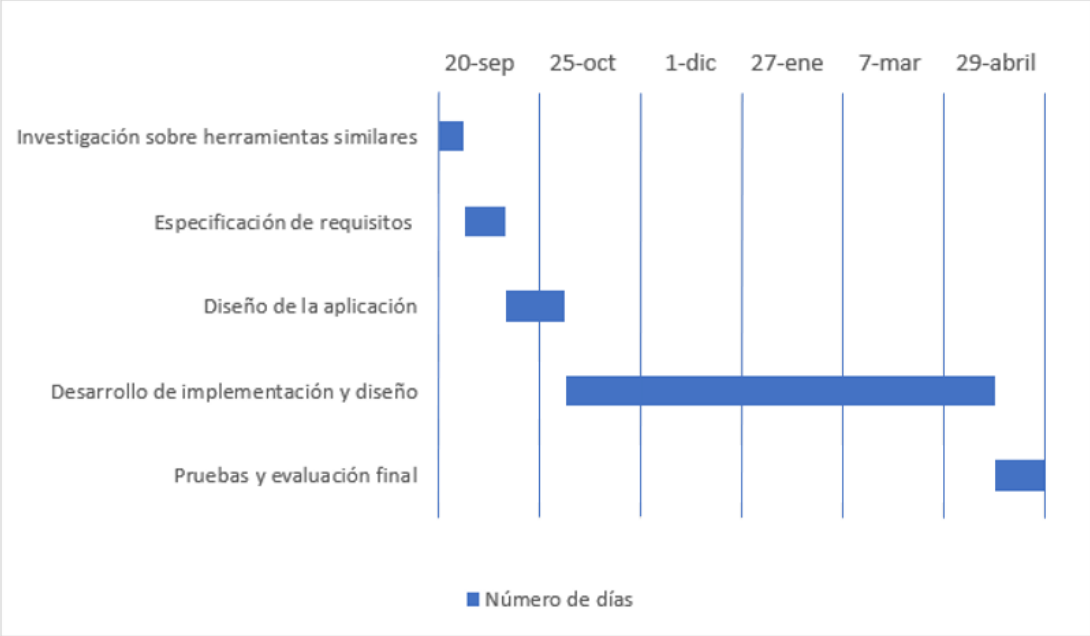


Figura 1: Planificación temporal

Capítulo 2: Estado del arte

En este capítulo, se va a presentar el estado del arte realizado para el desarrollo del proyecto. En concreto, se comentarán las técnicas y herramientas que suelen utilizarse para el análisis sintáctico de las oraciones. A continuación, se describirán las herramientas automáticas de análisis sintáctico, y herramientas de representación gráfica del resultado del análisis sintáctico. Por último se presentarán aplicaciones con funciones similares a las desarrolladas en este proyecto.

2.2. Técnicas de análisis sintáctico

El análisis sintáctico, es el proceso que determina las relaciones sintácticas entre los diferentes componentes que forman una oración, con el objetivo de observar la forma en la que se agrupan y las relaciones jerárquicas que existen entre los mismos [1].

2.2.1. Corpus lingüísticos

Un corpus lingüístico es una extensa colección de textos u oraciones orales transcritas, que muestran el uso real de una lengua, de la forma más exacta posible. [2]. Un corpus lingüístico tiene características propias, como son [3]:

- El número de textos que se pueden almacenar es muy amplio, y siempre finito.
- La información lingüística que se almacena es actualizable.
- En ellos se pueden ver como se relacionan las palabras que forman los textos.

En la práctica del desarrollo de corpus lingüísticos, pueden encontrarse corpus generales, y corpus especializados en una temática, dependiendo de los textos que posean en dichos corpus, algunos ejemplos de los temas de corpus especializados pueden ser jurídicos, médicos, informáticos, periodísticos...

Actualmente los corpus se desarrollan y almacenan de manera electrónica, es por ello que la utilización de un corpus lingüístico se ha convertido en una de las formas más eficaces y rápidas de trabajar con muestras textuales para, realizar un análisis sintáctico.

2.2.2. TreeBank

Es un tipo de corpus anotado, donde se guarda un conjunto de frases parseadas y etiquetadas gramaticalmente [1]. Además también se almacenan las relaciones sintácticas de las palabras [4]. La representación de estos corpus lingüísticos es mediante una estructura arbórea. Las características principales que debe poseer un corpus de tipo TreeBank son [5]:

- Guarda la información léxico-ortográfica de cada palabra de cada oración.
- Indica las dependencias entre las palabras que forman la oración.
- Representa de una forma arbórea de los componentes de cada oración, en base a sus dependencias.

La mayoría de los analizadores hacen uso de treebanks, con el fin de segmentar las palabras que conforman las oraciones, e investigar de elementos morfológicos y sintácticos para encontrar patrones y dependencias entre las palabras.

[1] Proceso para asignar a cada una de las palabras que forman un texto su categoría gramatical

2.2.3. Análisis basado en dependencias

El análisis basado en dependencias [2] es un proceso en el que se analiza la estructura sintáctica de una oración, y se comprueba que relaciones hay entre las palabras que la conforman. Con las dependencias entre las palabras encontradas, se podrá establecer un árbol sintáctico de dependencias, en base a la información sintáctica de cada uno de los componentes. Normalmente esta técnica es utilizada con ayuda de un corpus sintáctico orientado al texto. Las características básicas de esta técnica son:

- Permite la representación gráfica de las dependencias. Normalmente esta representación es en forma de árbol.
- Determina la importancia de las palabras que componen la oración, en base a su altura en el árbol sintáctico de dependencias.
- Encuentra cuál será la acción principal de la oración. Esta será el verbo principal de la oración y constituirá la raíz del árbol mencionado anteriormente.

El empleo de esta técnica será de gran importancia. En primer lugar, en el análisis sintáctico y para la etiquetación semántica de las palabras de las oraciones. En segundo lugar, para determinar la importancia de las palabras en la oración, donde el verbo se tomará como centro estructural y todos los demás componentes como nodos. Finalmente se llevará a representar una estructura arbórea con todos los componentes [7].

2.3. Anotación de información del análisis sintáctico

2.3.1. Anotación sintáctica

Es el proceso de asignar la información morfológica y la anotación gramatical a cada una de las palabras que componen un texto [3].

Al realizar el etiquetado, se utilizan diferentes reglas que predicen las posibles categorías de una palabra. La principal desventaja de esta técnica es el gran coste de tiempo para implementar las reglas y la exclusividad de dichas reglas a una lengua concreta.

Una técnica alternativa al desarrollo manual del etiquetado, es el uso de aprendizaje automático, mediante un corpus lingüístico. El objetivo es construir un modelo estadístico, que obtiene la etiqueta gramatical de cada palabra en base a la probabilidad sobre diferentes posibles etiquetados. Un ejemplo de esta técnica es el sistema CLAWS [8], el cual presenta una probabilidad de acierto de más del 90 por ciento. La principal desventaja de estas técnicas es el alto tiempo que se necesita para determinar la categoría correcta, debido a que se analizan todos los posibles etiquetados potenciales de una palabra [9].

2.3.2. El formato ConLL

El formato CoNLL[10] es un formato para representar la anotación sintáctica de las palabras que componen las oraciones. Permite mantener información acerca de las dependencias sintácticas de las palabras que conforman las oraciones.

[2] Se entenderá dependencia, dentro de la unidad lingüística, como la conexión entre las palabras que conforman una frase u oración [6].

[3] Algunas palabras pueden pertenecer a varias categorías gramaticales dependiendo del contexto de la oración y/o la ambigüedad de las palabras

Este formato ofrece soporte para varios idiomas [11], y existen varias versiones del formato dependiendo de la información que se desee almacenar [4]. La información representada se caracteriza por [12]:

1. La información semántica/sintáctica de cada palabra se almacena en una sola línea por lo que la información de una oración se almacena en una o más líneas.
2. Los archivos en formato ConLL son archivos de texto plano en con formato codificado UTF-8.
3. El final de información de una línea de un archivo ConLL se representa como un salto de página.
4. En un archivo CoLL existen 3 tipos de líneas:
   - Líneas con la información de cada palabra, con una serie de campos de información separados por tabuladores. En caso de que algún campo sea vacío se indicará con el signo "_".
   - Líneas vacías en blanco para determinar el final de una oración y el principio de la siguiente.
   - Líneas que comienzan con el signo "#" que representan comentarios.

Los principales campos de información utilizados en ConLL-U (ConLL Universal) son [12]:

- **Id:** identificador numérico de una palabra en la oración. Representa el orden de dicha palabra en la proposición. Empieza en 1 por cada nueva oración.
- **Forma:**palabra de la oración o símbolo de puntuación.
- **Lema:** lema o raíz de la palabra.
- **UPOS:** etiqueta universal que marca las categorías principales del discurso tal como ADJ(adjetivo), ADV(adverbio), NOUN(sustantivo).
- **XPOS:** etiqueta que especifica el idioma. Dicho campo puede aparecer con el símbolo "_" al no ser obligatorio.
- **Características:** lista de características morfológicas de cada palabra tal como número, género, tiempo, etc. Cada característica irá separada por el signo "—". En singular el tipo de característica especificada dependiendo del tipo de palabra (no es lo mismo un sustantivo, verbo, adjetivo...). Puede aparecer con el símbolo "_" al no ser obligatorio.
- **Cabecera:** índice del padre semántico, (es decir referencia a la palabra que depende esa palabra desde el punto de vista sintáctica). Es un valor de Id mayor o igual a 0, en caso de ser el root (raíz) de la oración.
- **Deprel:** representa la relación sintáctica entre la cabecera y esa palabra. Existen 37 posibles relaciones sintácticas universales tal como como, *nsubj ,nmod, advcl...*
- **Deps:** gráfico de dependencias en forma de lista de tipo cabecera-deprel.
- **Miscelánea:** en este campo pueden aparecer otro tipo de anotaciones. En caso de que este campo fuera irrelevante aparecerá el símbolo de campo vacío "_".

[4] La diferencia principal entre las diferentes versiones ConLL se encuentra en los campos de información que se utilizan para representar la información

En la Figura 2 se muestra un ejemplo de tres líneas en formato ConLL-U, donde se describe la información de tres palabras de una oración, de acuerdo con los campos listados anteriormente.

```
1  El        el    DA0MS0  DA  pos=determiner|type=article|gen=masculine|num=singular 2 det   (grup-verb:3(sn:2(espec-ms:1(j-ms:1)) -
2  ministro  ministro NCMS000 NC  pos=noun|type=common|gen=masculine|num=singular       3 nsubj (grup-nom-ms:2(n-ms:2)))             -
3  afirmó    afirmar  VMIS3S0 VMI pos=verb|type=main|tense=past|person=3|num=singular   0 root  (verb:3)                             -
```

Figura 2: Ejemplo de oración con formato CoNLL-U

Por ejemplo el significado de la línea 1 del ejemplo de la Figura 2 significa:

- **1:** corresponde con el campo id, e identifica la posición de la palabra en la oración.
- **El:** representa la palabra real.
- **el:** raíz de la palabra, es la parte de la palabra que no varía y que indica su significado principal.
- **DA0MS0 y DA:** corresponden a XPOS y UPOS respectivamente. Su función es la de poner una etiqueta a la palabra para categorizarla.
- **pos=determiner—type=article—gen=masculine—num=singular:** describe el campo categoría. Se puede ver toda la información morfológica de la palabra, en este caso "el" es un determinante artículo, con género masculino y singular.
- **2:** identificador de la palabra que depende. En este caso "El" depende de la palabra con id=2, es decir "ministro". Así mismo esta última, depende de la palabra con id=3, "afirmó". Finalmente esta última al ser 0 su dependiente, indica que es el root de la oración. Por tanto en esta oración, "afirmó" es el padre , "ministro" es su hijo, y "El" es el hijo de "ministro".
- **det:** indica la relación que tiene con el dependiente. En este caso la relación que tiene "El", con "ministro" es que es determinante.
- **(grup-verb:3(sn:2(espec-ms:1(j-ms:1)):** corresponde al campo deps. Se muestra una representación de las dependencias, entre las palabras de la oración.
- **-:** Último campo, es utilizado para anotaciones varias. En este caso es vacío por lo que se representa con el signo "-", para indicarlo.

2.4. Herramientas para el análisis sintáctico

En esta sección se va a describir un conjunto de herramientas que permiten realizar el análisis sintáctico automático de oraciones.

2.4.1. Freeling

Es un proyecto de código abierto, desarrollado por la Universidad Politécnica de Cataluña [13], que permite realizar el análisis sintáctico de oraciones. Está implementado como una librería de C++, y puede ser utilizado desde cualquier aplicación, además, ofrece soporte para múltiples idiomas como español, inglés, italiano, francés, ruso, etc.

Las principales funcionalidades que ofrece esta herramienta son las siguientes: análisis morfológico, análisis sintáctico, detección de entidades con nombre, desambiguación de las palabras

y etiquetado semántico de la categoría gramatical a la que corresponda cada palabra.

En el caso del análisis sintáctico devuelve como resultado la anotación sintáctica de los textos analizados en un archivo en formato XML, JSON o ConLL.

2.4.2. CoNLL-U Parser

Es un analizador sintáctico desarrollado en Python que permite extraer las oraciones de un archivo con formato CoNLL-U [14]. Sus principales características son:

1. Tiene la posibilidad de ser añadido, como una librería a Python, para poder usar las funciones.
2. Ofrece la posibilidad de guardar la oración extraída del fichero ConLL en formato de texto plano, o en estructuras de datos, tales como como diccionarios, listas o arrays.
3. Puede convertir la oración obtenida del fichero leído, a una estructura arbórea.
4. Permite filtrar campos (id,lema,forma...).

2.5. Herramientas de representación gráfica del resultado del análisis sintáctico

Tras realizar el análisis sintáctico de un texto, el siguiente paso es la representación de forma gráfica de la oración, con el fin de observar cómo se interconectan los componentes que forman el texto.

A continuación se presentará un conjunto de herramientas orientadas a la representación gráfica del resultado del análisis sintáctico y en algunos casos de la manipulación de dicha representación.

2.5.1. Grew

Es una herramienta de procesamiento de textos, búsqueda de patrones para la simplificación de palabras en una oración y representación gráfica en forma de árboles de dependencia. Para dicha representación, esta herramienta utiliza las etiquetas semánticas, y las dependencias que existan entre las palabras del corpus escogido. Su última versión 1.3, fué lanzada el 24 de junio de 2019 [15]. La herramienta puede ser usada desde la página web, o bien se puede instalar de manera local, en diferentes entornos como Linux, Mac OS X o Windows. Grew posee dos tipos de uso:

- **Grew-match:** permite al usuario crear y probar patrones de búsqueda en uno de los corpus lingüísticos que están almacenados en la aplicación [16]. El usuario describe los patrones usando la notación Ocaml [17] en el editor de textos que posee la herramienta. Posteriormente se visualizará el gráfico de dependencias entre las palabras que conforman la oración, en base al patrón encontrado, que cumplan la propiedad propuesta. En el gráfico que se visualiza, cada nodo representa las palabras de la oración y cada arista representa las dependencias que hay entre los mismos.

- **Grew-parse:** el usuario podrá escribir una frase, y la herramienta la representará en un grafo de las dependencias de las palabras que haya en la oración escrita. Mostrando además la información morfológica de cada palabra. Actualmente solo está disponible para ser probados con textos en francés[18].

Si se desea buscar patrones, pueden buscarse en sus bancos de árboles. En la actualidad Grew contiene más de 130 bancos de árboles [19]. La forma en la que se almacena la información en estos árboles, es mediante el formato ConLL-U. A continuación en la Figura 3, se muestra un ejemplo de búsqueda de un patrón, el cuál buscará en el banco de datos seleccionado, un nodo vacío que sea además un verbo.

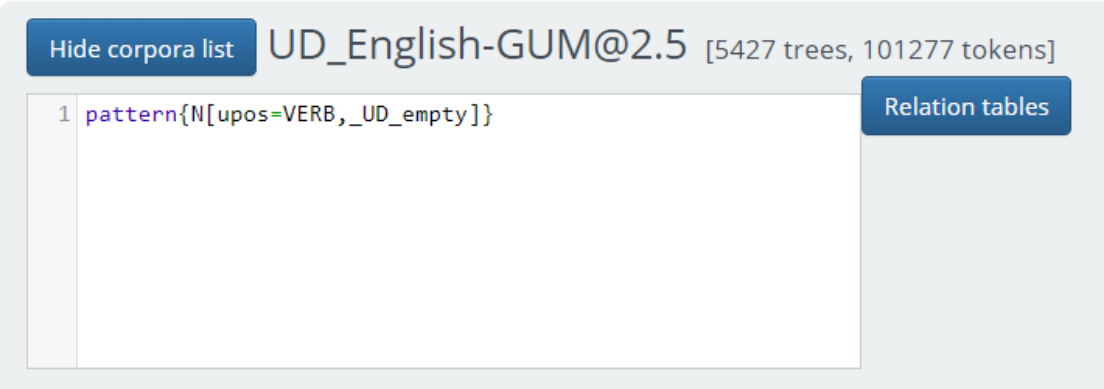


Figura 3: Ejemplo de patrón de búsqueda en la herramienta Grew-match

2.5.2. Tred

Es una aplicación utilizada para el procesamiento de corpus lingüísticos y la visualización y manipulación de gráficos en forma de árbol[20]. En estos gráficos se podrán visualizar la información morfológica de cada palabra del corpus, y las dependencias encontradas, donde cada nodo corresponderá a cada palabra del corpus. La herramienta está desarrollada en el lenguaje de programación Perl y permite su ejecución en los principales sistemas operativos Windows, OS o Linux.

2.5.3. DgAnnotator

Es una biblioteca desarrollada en lenguaje Java y compatible con cualquier sistema operativo[21]. La función de esta herramienta es el análisis textual, realizando un etiquetado gramatical de las palabras para la creación un árbol de dependencias. En la visualización de este árbol, se resaltan los nodos con diversos colores con el objetivo de ser más visuales e intuitivos.

Los resultados del análisis se proporcionan en un archivo con formato XML, CoNLL-X y CoNLL-U. Además cuenta con la posibilidad de descargar el árbol resultante del análisis, en un archivo con formato PNG.

2.6. Proyectos similares

A continuación se describen un conjunto de aplicaciones web y bibliotecas que realizan funciones similares a las de este proyecto.

2.6.1. Treex

Es una herramienta de procesamiento de lenguaje natural, desarrollado con el lenguaje de programación Perl[22].

La forma de utilizar esta herramienta es mediante una interfaz para que el usuario pueda cargar un archivo, con extensión *.treex*. Dicha extensión es propia de la herramienta, pero también acepta y genera archivos con formato CoNLL-U, ayudando así a mantener los estándares de almacenamiento de la información del análisis de texto. Otra alternativa es introducir una ruta web y analizar el contenido textual mediante los corpus que se tienen almacenados[23].

Además incluye una API para comunicarse con la herramienta para la manipulación de árboles de dependencias. Contando con soporte para español, inglés y checo.

2.6.2. Brat

Es una herramienta utilizada para mostrar en una estructura con forma de árbol, las palabras de forman una oración, la información morfológica de cada palabra y la forma que tienen de interconectan las dependencias entre las palabras de la oración[24].

Utiliza diversos colores para resaltar las categorías gramaticales a las que pertenecen cada palabra, y para las dependencias que existan entre las palabras utiliza fechas, que se muestra la dependencia por la que se unen las palabras.

2.6.3. UDPipe

Es una plataforma utilizada para el análisis, etiquetado y lematización del contenido de un archivo con formato formato CoNLL-U, tomando como soporte un corpus lingüístico utilizado por el usuario.

Esta herramienta puede ser entrenada con los datos del fichero ConLL cargado, para mejorar el analizador morfológico y lematizador [25]. Dicha aplicación posee diversos ejemplos de modelos entrenados para cualquier corpus semántico.

UDPipe, se puede utilizar en su aplicación web, y como biblioteca para C++, python, Java, Perl y C.

2.6.4. Deep search

Es una herramienta, desarrollada por la Universidad de Stanford[26], que permite realizar búsquedas de patrones de expresiones sintácticas en una representación en forma de árbol de la estructura sintáctica de un texto. Los patrones de búsqueda como expresiones regulares representadas usando el lenguaje Tsurgeon [5] o bien el lenguaje Semgrex [6][27].

Presenta una interfaz[28], donde el usuario puede escribir un patrón mediante expresiones regulares para realizar la búsqueda en el árbol.

2.6.5. PyConll

Es una biblioteca desarrollada en Python que permite la manipulación de archivos en formato ConLL[31]. Sus principales funciones son:

- Permite la lectura de ficheros CoNLL y el almacenamiento de su contenido.
- Transforma y analiza de los datos cargados para serializar los componentes con formato CoNLL-U.

[5] Lenguaje para la transformación de árboles en función de un conjunto de patrones con el árbol [29]

[6] Es un lenguaje para codificar patrones mediante expresiones regulares y hacerlos coincidir en el gráfico, también permite operar los grafos mediante la consulta de arcos y de nodos [30]

- Reasigna las etiquetas semántica a cada palabra que forma la oración.

Capítulo 3: Tecnologías empleadas

En el presente capítulo se describen las tecnologías utilizadas para desarrollar el proyecto, las cuales se pueden agrupar en: tecnologías utilizadas para la interfaz del usuario, las usadas para la lógica del proyecto y otras herramientas.

3.2. Tecnologías usadas para frontend

La interfaz de usuario de la aplicación web ha sido implementada mediante tecnologías pensadas para el desarrollo web, como HTML5, CSS3, Boostrap .

A continuación se desarrollará cada una de las tecnologías empleadas.

3.2.1. HTML5

Es un lenguaje marcado, utilizado para la creación de páginas webs[34]. Ofrece un gran abanico de posibilidades para diseñar diversos elementos en la web, pudiendo combinarse con diferentes lenguajes como PHP, Javascript, CSS o Python.

HTML5 destaca también la compatibilidad de sus elementos con la mayoría de los navegadores, como Google Chrome, Firefox, Safari... Lo que resulta una herramienta óptima para la implementación de las páginas web. Otra característica muy útil de HTML5, es su capacidad para utilizar numerosas APIs, con el fin ayudar en el desarrollo de complejas funcionalidades en las aplicaciones web. Más concretamente, la API de integración con DOM[7] y la buena integración con otro tipo de tecnologías como CSS3, PHP o Javascript.

3.2.2. CSS3

Es un lenguaje que permite definir el estilo de cualquier elemento de una página web, con el fin de personalizar la apariencia de una web, como la tipografía utilizada, los márgenes de la página, las imágenes, etc[36].

Además esta tecnología puede complementarse con Bootstrap[8] mediante plantillas y diferentes componentes, con el fin de mejorar la interfaz mostrada al usuario para que fuera más atractiva y visual.

3.2.3. Bootstrap

Es una biblioteca de código abierto para diseñar aplicaciones web. Tiene plantillas para el diseño de tipografías, formularios, botones o menús de navegación. Todos estos elementos diseñados mediante HTML, CSS y JavaScript. Además todos los componentes con lo que cuenta Bootstrap, son responsive, y se ajustarán a los diferentes tamaños de pantalla automáticamente.

3.3. Tecnologías usadas para backend

Las tecnologías que se han usado para el desarrollo lógico de la aplicación han sido:

[7] Modelo de Objetos del Documento[35], utilizado para la representación de documentos HTML,XML. Con DOM, se puede acceder dinámicamente a los objetos compuestos en estos documentos y manipularlos, por ejemplo en su estilo, estructura o contenido

[8] Framework, de código abierto, para conseguir un diseño responsive, para ajustar la página web a diferentes dimensiones de pantalla.

3.3.1. PHP

Es un lenguaje para la comunicación con el servidor y con la base de datos[37]. Muy utilizado en el desarrollo web ya que permite utilizarse con HTML. Este lenguaje se distingue de otros lenguajes de script como JavaScript, ya que este se ejecuta del lado del servidor y se enviará al cliente el resultado del script.

PHP puede ser utilizado en la mayoría de sistemas operativos y servidores. Además, permite una programación orientada a objetos o a procedimientos. Otra característica muy importante de este lenguaje es que se puede trabajar con múltiples bases de datos diferentes y realizar operaciones en ellas.

3.3.2. JavaScript

Es un lenguaje de programación interpretado, utilizado principalmente en su forma del lado del cliente como parte de un navegador web, permitiendo incorporar diversas funcionalidades dinámicas para que sea visualizadas por el usuario[38].

Javascript es altamente compatible con muchos lenguajes y puede ser integrado con HTML y PHP. Además con esta tecnología tiene numerosas funciones, librerías y APIs, permitiendo mejoras en la interfaz convirtiéndolas en páginas dinámicas.

3.3.3. JQuery

Es una biblioteca de JavaScript que permite simplificar la forma interactuar con los componentes utilizados en HTML, manipular el árbol DOM[9] y el manejo de eventos de una manera más simple[39]. Además cabe destacar el soporte que tiene para múltiples navegadores.

Uno de los usos puede ser modificar la estructura DOM de una manera más simple que utilizando únicamente JavaScript. Utilizando para ello las clases y eventos de los que dispone esta herramienta y la manipulación en la hoja de estilos CSS.

3.3.4. Python

Python es un lenguaje de programación interpretado, dinámico y multiplataforma, administrado por Python Software Foundation[40].

Ofrece varios estilos de programación como son: orientado a objetos, programación imperativa y programación funcional. Algunas características más importantes de python son[40]:

- Tiene una gran simplicidad para integrarse con otras herramientas como con PHP.
- Posee numerosas librerías externas con múltiples funciones y llamadas a diversas APIs.
- Lenguaje poco tipado, con el que se busca el desarrollo de cualquier tipo de programa.

[9] Modelo de Objetos del Documento[35], utilizado para la representación de documentos HTML,XML. Con DOM, se puede acceder dinámicamente a los objetos compuestos en estos documentos y manipularlos, por ejemplo en su estilo, estructura o contenido

3.3.5. SQL

Es un lenguaje de consulta estructurada, diseñado para administrar y recuperar la información de bases de datos relacionadas mediante consultas [43].

Es un lenguaje de alto nivel que explota la potencia de los sistemas relacionales, permitiendo un manejo de los registros. En su alcance, permite a los usuarios llevar a cabo tareas como selección, inserción, actualización y eliminación de los datos.

3.4. Otras herramientas

3.4.1. MySQL

Es un sistema de gestión para bases de datos relacionales [42]. MySQL permite la ejecución de transacciones de consultas y el uso de claves foráneas entre las tablas de la base de datos, y presenta un soporte en todos los sistemas operativos. Suele utilizarse para tratar con la base de datos y para la preparación de consultas ejecutadas en la base de datos.

3.4.2. PHPMyAdmin

Para poder acceder a la información de una base de datos relacional de tipo MySQL, se suele usar PHPMyAdmin.

Es una herramienta, escrita en PHP, destinada al manejo de la administración de MySQL en la web. A través de una interfaz, el usuario puede crear, modificar, o eliminar campos, tablas, bases de datos (mediante lenguaje SQL), administrar privilegios a usuarios, tablas y exportar datos en varios formatos [41].

3.4.3. JSON

Es un formato de texto para el envío de datos bidireccional [45] que se caracteriza por:

- No esta ligado a los lenguajes de programación.
- Es similar a una estructura de datos usada en los lenguajes de programación: registro, diccionario ...
- Los valores que se representan pueden ser de varios tipos: una lista ordenada de valores, o un conjunto de clave-valor.

Ya que todos los lenguajes de programación admiten estas dos estructuras, hacen que sea una buena opción utilizar este formato para el paso de mensajes.

3.4.4. Servidor Apache

Es un servidor web HTTP de código abierto para plataformas como Linux, Windows, o Macintosh, que implementa el protocolo HTTP[10] [44]. El uso principal de Apache es para mostrar las webs desarrolladas.

[10]HTTP es un protocolo basado en el principio de cliente-servidor. Cada petición se envía a un servidor, el cuál la gestiona y la responde.

Suele utilizarse para lanzar la aplicación a un servidor HTTP y comprobar su funcionamiento y visualizar su interfaz localmente.

3.4.5. Sublime Text 3

Es un editor de texto multiplataforma y simple[46]. Su interfaz de color negro, y el coloreado de las instrucciones, es concebida para resaltar mejor la sintaxis de programación. Además posee un minimapa, para visualizar el contenido del archivo de una manera más reducida, con el fin de moverse más rápido por el código.

Capítulo 4: Desarrollo del proyecto

En este capítulo se explicará la metodología de trabajo seguida durante el desarrollo del proyecto, la especificación de requisitos del sistema, y se detallará cuál es la arquitectura de la aplicación y el modelo de datos utilizado.

4.2. Marco de trabajo

El marco de trabajo escogido para el desarrollo ágil del proyecto, ha sido Scrum [32]. Los roles usados durante el desarrollo del proyecto han sido fieles a la teoría de Scrum, donde había un *Scrum Master*, en este caso, fue el director del proyecto, que gestionaba las tareas y los tiempos entre los sprint. Dos *Product Owner*, que han sido las directoras del proyecto. Y el *Team*, pero con una adaptación, ya que esta metodología, está pensada para el trabajo colaborativo, y este proyecto ha sido realizado de una manera individual, por lo que el equipo ha sido de una persona.

En cuanto a los *sprint*, tuvieron al comienzo del desarrollo una duración de 3 semanas, y en la última etapa del proyecto una duración de 1 semana. Todos los sprint desarrollaban incrementalmente las funciones requeridas. Se puede agrupar todos estos sprints en cinco iteraciones principales:

1. En la primera iteración, se elaboró el proceso de lectura de corpus lingüísticos con formato .CoNLL y almacenamiento de información. También se añadió la posibilidad de leer archivos que no tuvieran ese formato.

2. En la segunda iteración, se desarrolló un formulario para que el usuario pudiera crear reglas, aplicarlas a las oraciones y así simplificarlas.

3. En la tercera iteración, se creó un visualizador de oraciones, representado como un gráfico con forma de árbol, para que los usuarios pudieran visualizar las palabras de las oraciones y la información morfológica de todas ellas.

4. En la cuarta iteración, se desarrolló el editor de patrones, para que los usuarios pudiesen escribir patrones de búsqueda en el corpus cargado, utilizando para ello la herramienta Grew, y se unió con el visualizador de árboles comentado en la anterior interacción.

5. En la quinta iteración, se añadió la posibilidad de que el usuario pudiera escribir reglas lingüísticas para que las aplicase en el corpus cargado mediante la utilización de Grew. Y con ello simplificar las oraciones compuestas. A continuación se añadió al visualizador de árboles anteriormente comentado.

4.3. Especificación de requisitos

En esta sección se desarrollará la especificación de los requisitos del sistema. Para ello se explicarán quienes son los actores que se comunican con el sistema y los casos de uso que se han implementado.

4.3.1. Actores

Los actores que se han determinado son:

- **Usuario**: este actor representa el usuario que utilizará la herramienta.
- **Sistema**: este actor será el encargado de cargar los corpus lingüísticos, los analice, interprete los patrones y reglas escritas y muestre los resultados.

4.3.2. Casos de uso

A continuación en la Figura 4 se presenta el diagrama de casos de uso, en el que se muestra todas las funcionalidades que tendrá el usuario y el sistema.

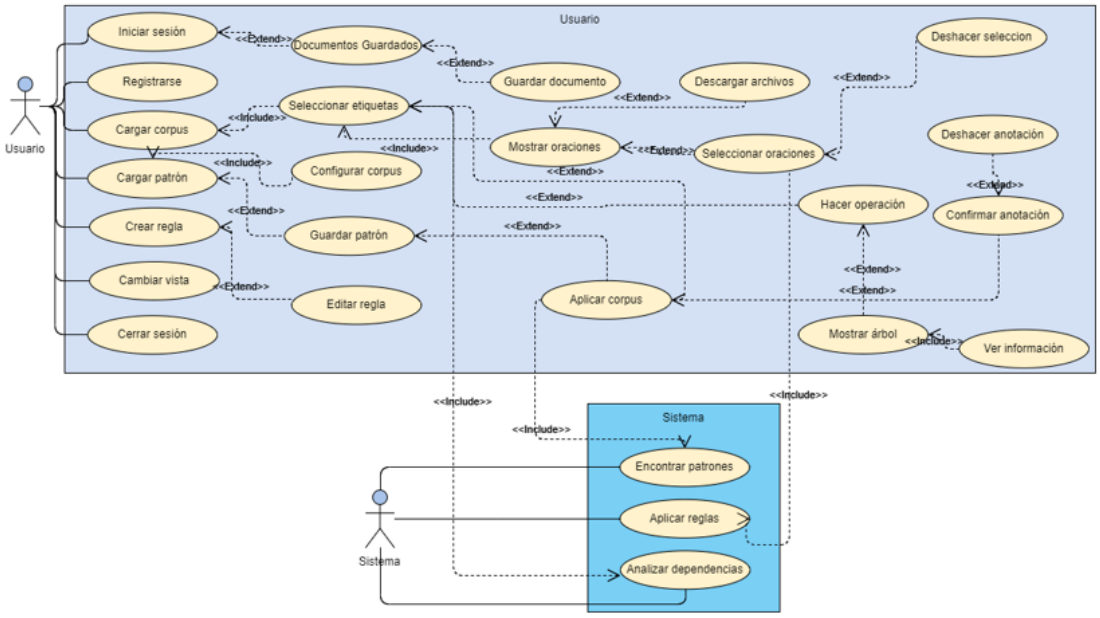


Figura 4: Casos de uso

4.3.2.1 Casos de uso del usuario

UC-0001	Iniciar sesión	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El usuario podrá iniciar sesión en el sistema y tener una sesión activa	
Precondición	El usuario desea iniciar sesión en la aplicación	
Secuencia normal	Paso	Acción
	1	El usuario pulsa en el botón de iniciar sesión de la aplicación
	2	El usuario introduce su usuario y contraseña
	3	El sistema valida los datos insertados por el usuario
	4	Los datos introducidos por el usuario son válidos e inicia sesión
	5	El usuario tendrá la opción de ver todo al igual que un usuario que no haya iniciado sesión y además los documentos guardados **UC-0008**
Postcondición	El usuario ha iniciado sesión en la aplicación	
Excepción	Paso	Acción
	1	Los datos introducidos por el usuario son incorrectos, se notifica al usuario del error
Importancia	No vital	
Estabilidad	Alta	

Figura 5: Funcionalidad 1 - Iniciar sesión

UC-0002	Registrarse	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El usuario podrá registrarse en la aplicación	
Precondición	El usuario no está registrado en el sistema	
Secuencia normal	Paso	Acción
	1	El usuario pulsa en el botón de registrarse de la aplicación
	2	El usuario introduce los datos necesarios para registrarse
	3	El sistema valida los datos insertados por el usuario
	4	Los datos introducidos por el usuario son válidos se registra en el sistema
	5	El usuario tendrá la opción de ver todo al igual que un usuario que no haya iniciado sesión y además los documentos guardados **UC-0008**
Postcondición	El usuario se registra en la aplicación	
Excepción	Paso	Acción
	1	Los datos introducidos por el usuario son incorrectos, ya existe una cuenta con ese usuario o las contraseñas no coinciden, se notifica al usuario del error
Importancia	No vital	
Estabilidad	Alta	

Figura 6: Funcionalidad 2 - Registrarse

UC-0003	Cargar corpus	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El usuario podrá cargar un archivo que contenga un corpus lingüístico para analizarlo	
Precondición	El usuario selecciona un archivo con extensión *". conll"*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en el botón de cargar corpus lingüístico
	2	El sistema hace la lectura del contenido del archivo
	3	El sistema almacena toda la información leída del archivo
	4	Se notifica al usuario de que se han leído todos los datos correctamente, y continua el **UC-0009**
Postcondición	El usuario carga un corpus lingüístico con todos los textos que desea analizar	
Excepción	**Paso**	**Acción**
	1	El contenido del corpus tiene caracteres extraños y conlleva a una lectura errores del fichero.
Importancia	vital	
Estabilidad	Alta	

Figura 7: Funcionalidad 3 - Cargar corpus

UC-0004	Cargar patrón	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	Ninguno	
Descripción	El usuario podrá cargar un archivo con el patrón lingüístico que desea analizar	
Precondición	El usuario selecciona un archivo que tenga guardado un patrón codificado	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en el botón de cargar patrón lingüístico
	2	El sistema hace la lectura del contenido del archivo
	3	Se notifica al usuario de que se han leído todos los datos correctamente
	4	Se carga el contenido del fichero en la pantalla, para que el usuario pueda visualizar el patrón leído, con la posibilidad de editarlo
Postcondición	El usuario carga un patrón lingüístico en el editor que aparece en pantalla, podrá visualizarlo y editarlo	
Excepción	**Paso**	**Acción**
	1	El contenido del corpus tiene caracteres extraños y conlleva a una lectura errores del fichero.
Importancia	Vital	
Estabilidad	Alta	

Figura 8: Funcionalidad 4 - Cargar patrón

UC-0005	Crear regla	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario ha debido iniciar sesión o registrarse en la aplicación para crear reglas **UC-0001** o **UC-0002** y ha cambiado de vista, **UC-0006**	
Descripción	El usuario podrá crear una regla a partir de un formulario con varias opciones	
Precondición	El usuario inicia sesión y quiere crear una regla lingüística	
Secuencia normal	**Paso**	**Acción**
	1	El usuario completa los datos del formulario de creación de reglas
	2	El sistema crea la regla y se añaden en la base de datos a las reglas creadas de ese usuario
	3	Se notifica al usuario de que se ha creado correctamente
Postcondición	El usuario crea la regla lingüística y se visualiza la nueva regla en la lista de reglas que se presentan en pantalla al usuario	
Excepción	**Paso**	**Acción**
	1	El usuario no ha iniciado sesión, por lo que se notifica al usuario de que no puede crear reglas.
Importancia	No vital	
Estabilidad	Alta	

Figura 9: Funcionalidad 5 - Crear regla

UC-0006	Cambiar vista	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	Ninguna	
Descripción	El usuario podrá cambiar el tipo de interfaz que está visualizando	
Precondición	El usuario pulsa en el botón de cambiar vista	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa la opción de cambiar vista
	2	El sistema modifica la vista del usuario. Si estaba en la vista avanzada cambia a la básica y viceversa
Postcondición	El usuario puede ver la nueva vista	
Excepción	**Paso**	**Acción**
	-	-
Importancia	Vital	
Estabilidad	Alta	

Figura 10: Funcionalidad 6 - Cambiar vista

UC-0007	Cerrar sesión	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario ha iniciado sesión o se ha registrado en la aplicación **UC-0001** o **UC-0002**	
Descripción	El usuario podrá cerrar la sesión abierta	
Precondición	El usuario había iniciado sesión	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa el botón de cerrar sesión
	2	El sistema cierra la sesión abierta
	3	Se notifica al usuario de que se ha cerrado sesión
Postcondición	El usuario podrá ver de nuevo las opciones para iniciar sesión y registrarse	
Excepción	**Paso**	**Acción**
	-	-
Importancia	No vital	
Estabilidad	Alta	

Figura 11: Funcionalidad 7 - Cerrar sesión

UC-0008	Documentos guardados	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe iniciar sesión para poder visualizar esta opción, debe haber completado **UC-0001** o **UC-0002**	
Descripción	El usuario podrá observar y descargar todos los documentos que haya guardado durante la utilización de la herramienta	
Precondición	El usuario ha iniciado sesión	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa la opción de documentos guardados
	2	El sistema redirige al usuario a la página donde tiene almacenados todos los archivos guardados, y visualiza todos los documentos que ha guardado, tanto su nombre, fecha y con la opción de descargar cualquiera de ellos
Postcondición	El usuario puede ver todos los documentos que ha guardado	
Excepción	**Paso**	**Acción**
	-	-
Importancia	No vital	
Estabilidad	Alta	

Figura 12: Funcionalidad 8 - Documentos guardados

UC-0009	Seleccionar etiquetas	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber cargado un corpus lingüístico, haber completado **UC-0003**	
Descripción	Una vez el usuario haya cargado un corpus, el usuario deberá seleccionar las etiquetas verbales que haya en el corpus cargado.	
Precondición	El usuario pulsa la opción de seleccionar etiquetas	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa la opción de seleccionar etiquetas verbales
	2	El sistema muestra las 15 primeras oraciones, con toda la información morfológica de las palabras.
	3	El usuario escribe las etiquetas verbales que tiene el corpus y finalizará
	4	Se notificará al usuario de que se ha cargado el corpus con las etiquetas escritas correctamente
Postcondición	El usuario ha seleccionado todas las etiquetas verbales que hay en el corpus.	
Excepción	**Paso**	**Acción**
	1	No se haya ninguna etiqueta verbal que ha escrito el usuario. Se guardarán las etiquetas incorrectamente.
Importancia	Vital	
Estabilidad	Alta	

Figura 13: Funcionalidad 9 - Seleccionar etiquetas

UC-0010	Configurar corpus	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber cargado un corpus lingüístico, haber completado **UC-0003**	
Descripción	Una vez el usuario haya cargado un corpus, si el contenido del fichero no es el esperado, se preguntará al usuario donde está cada elemento necesario en el archivo para que se ajuste a un formato CoNLL-U	
Precondición	El usuario ha cargado el corpus lingüístico	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa la opción de seleccionar columnas
	2	El sistema muestra la primera oración cargada para que el usuario pueda ver como se disponen las palabras en la oración
	3	El usuario deberá completar el formulario que aparecerá preguntando donde está cada elemento necesario en el archivo.
	4	Se notificará al usuario de que se ha cargado el corpus correctamente
Postcondición	El usuario ha completado todo el formulario y se ha cargado el corpus	
Excepción	**Paso**	**Acción**
	1	El usuario completa el formulario, pero no correctamente y al cargar el contenido del archivo, se cargan elementos por error.
	2	Se notifica al usuario con el error acontecido
Importancia	Vital	
Estabilidad	Alta	

Figura 14: Funcionalidad 10 - Configurar corpus

UC-0011	Guardar patrón o regla	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber cargado o escrito un patrón lingüístico, haber completado **UC-0004**	
Descripción	Una vez el usuario haya cargado un patrón/regla o lo haya escrito en el editor disponible, al pulsar en guarda, el patrón/regla se guardará hasta ese momento	
Precondición	El usuario ha cargado o escrito el patrón lingüístico	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa la opción de guardar patrón/regla
	2	El usuario deberá escribir como quiere llamar al patrón/regla guardado
	3	El sistema guardará el patrón/regla escrito con el nombre propuesto
Postcondición	El usuario ha escrito un patrón/regla y lo ha guardado en el sistema	
Excepción	**Paso**	**Acción**
	-	-
Importancia	Vital	
Estabilidad	Alta	

Figura 15: Funcionalidad 11 - Guardar patrón

UC-0012	Editar regla	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber creado una regla, completado **UC-0005** y debe iniciar sesión o estar registrado en la aplicación, **UC-0001 o UC-0002**	
Descripción	El usuario puede editar una regla que haya creado anteriormente, se guardarán los cambios propuestos	
Precondición	El usuario selecciona una regla	
Secuencia normal	**Paso**	**Acción**
	1	El usuario modifica los campos de la regla
	2	El usuario pulsa la opción de guardar cambios
	3	El sistema guardará los cambios en la base de datos correspondiente al usuario. Se notificará al usuario de que se ha editado la regla
Postcondición	El usuario ha editado una regla de su batería de reglas que había creado	
Excepción	**Paso**	**Acción**
	-	-
Importancia	No vital	
Estabilidad	Alta	

Figura 16: Funcionalidad 12 - Editar regla

UC-0013	Guardar documento	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe iniciar sesión o estar registrado en la aplicación, **UC-0001 o UC-0002**	
Descripción	El usuario podrá guardar los resultados de una oración analizada, en sus documentos guardados.	
Precondición	El usuario desea guardar una oración analizada	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en la opción de guardar documento
	2	El usuario deberá poner un nombre al documento para guardarse
	3	El sistema guardará el documento con el nombre propuesto en los documentos guardados de ese usuario.
	4	Se notificará al usuario que se ha guardado correctamente
Postcondición	El usuario ha guardado una oración en sus documentos guardados	
Excepción	**Paso**	**Acción**
	-	-
Importancia	No vital	
Estabilidad	Alta	

Figura 17: Funcionalidad 13 - Guardar documento

UC-0014	Mostrar oraciones	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber cargado el corpus y haber seleccionado las etiquetas verbales, **UC-0003, UC-0009**	
Descripción	Se mostrarán todas las oraciones al usuario que se hayan cargado del corpus	
Precondición	El usuario podrá visualizar las oraciones del corpus cargado	
Secuencia normal	**Paso**	**Acción**
	1	El usuario podrá visualizar todas las oraciones
	2	Si el usuario pulsa en cualquier oración podrá visualizar toda la información morfológica de cada palabra compuesta en la oración
Postcondición	El usuario visualiza toda la información de las palabras del corpus	
Excepción	**Paso**	**Acción**
	-	-
Importancia	Vital	
Estabilidad	Alta	

Figura 18: Funcionalidad 14 - Mostrar oraciones

UC-0015	Aplicar patrón al corpus	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber cargado el corpus y seleccionado las etiquetas verbales, **UC-0003, UC-0009** y haber guardado el patrón escrito **UC-0011**	
Descripción	Se aplicará el patrón lingüístico escrito al corpus cargado.	
Precondición	El usuario pulsará la opción de aplicar el patrón al corpus.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsará la opción de aplicar patrón
	2	El sistema analizará el corpus y buscará el patrón propuesto.
	3	El sistema devolverá las oraciones en las que se cumple ese caso escrito por el usuario, para ello se completará **UC-0024**. Y se notificará de éxito
Postcondición	El usuario habrá aplicado un patrón de búsqueda al corpus	
Excepción	**Paso**	**Acción**
	1	No se ha encontrado el patrón propuesto por el usuario en el corpus cargado. Se notificará al usuario.
Importancia	Vital	
Estabilidad	Alta	

Figura 19: Funcionalidad 15 - Aplicar corpus

UC-0016	Descargar archivos	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber cargado el corpus y seleccionado las etiquetas verbales, **UC-0003, UC-0009** y haber mostrado las oraciones **UC-0014**	
Descripción	El usuario podrá descargar las oraciones que desee una vez los haya seleccionado las oraciones	
Precondición	El usuario ha pulsado uno o varias oraciones para descargar	
Secuencia normal	**Paso**	**Acción**
	1	El sistema descargará las oraciones en formato. conll
	2	El sistema notificará al usuario de que se han descargado con éxito
Postcondición	El usuario habrá descargado las oraciones seleccionadas	
Excepción	**Paso**	**Acción**
	-	-
Importancia	Vital	
Estabilidad	Alta	

Figura 20: Funcionalidad 16 - Descargar archivos

UC-0017	Seleccionar oraciones	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber cargado el corpus y seleccionado las etiquetas verbales, **UC-0003, UC-0009** y haber mostrado las oraciones **UC-0014**	
Descripción	El usuario podrá seleccionar todas las oraciones que deseé	
Precondición	El usuario ha pulsado uno o varias oraciones de la lista de oraciones mostradas	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsará las oraciones que quiere seleccionar
	2	El usuario podrá seleccionar todas, una o ninguna
Postcondición	El usuario tendrá marcadas las oraciones interesadas	
Excepción	**Paso**	**Acción**
	-	-
Importancia	Vital	
Estabilidad	Alta	

Figura 21: Funcionalidad 17 - Seleccionar oraciones

UC-0018	Deshacer selección	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber completado **UC-0017**	
Descripción	El usuario deshará la selección propuesta de oraciones	
Precondición	El usuario ha pulsado uno o varias oraciones	
Secuencia normal	**Paso**	**Acción**
	1	El usuario desmarcará las oraciones que no quiera seleccionar
	2	Podrá desmarcar todas, una o ninguna
	3	Si desmarca todas no podrá descargar las oraciones seleccionadas **UC-0016**
Postcondición	El usuario habrá descargado las oraciones seleccionadas	
Excepción	**Paso**	**Acción**
	-	-
Importancia	Vital	
Estabilidad	Alta	

Figura 22: Funcionalidad 18 - Deshacer selección

UC-0019	Aplicar regla a la oración seleccionada	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario ha cargado el corpus y seleccionado las etiquetas verbales, ha completado por tanto **UC-0003, UC-0009,** además deberá seleccionar una oración **UC-0017.** El usuario ha escrito y guardado la regla **UC-0011**	
Descripción	El usuario selecciona una sola oración del corpus cargado, y podrá observar la oración y el resultado de la oración en caso de que se haya aplicado la **UC-0015**	
Precondición	El usuario ha seleccionado una oración del corpus cargado	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en el botón de Aplicar regla a la oración seleccionada
	2	Se mostrará la oración al usuario, el número de verbos encontrados gracias a la etiqueta verbal que haya configurado el usuario en **UC-0009**
	3	El sistema aplicará la regla a la oración, para comprobar si se puede reducir el número de palabras de la oración
	4	Se mostrará al usuario la oración reducida y el número de verbos en caso de haber seguido la **UC-0015**
	5	Se mostrará al usuario los árboles de dependencias de ambas oraciones, la inicial y la reducida.
Postcondición	El usuario habrá descargado las oraciones seleccionadas	
Excepción	**Paso**	**Acción**
	1	No se puede aplicar la regla a la oración, porque no está bien formada o porque no se puede reducir la oración con dicha regla. Se mostrará un mensaje informativo al usuario
Importancia	Vital	
Estabilidad	Alta	

Figura 23: Funcionalidad 19 - Hacer operación

UC-0020	Mostrar árbol	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber completado **UC-0019**	
Descripción	El usuario podrá visualizar el árbol de dependencias de la oración seleccionada	
Precondición	El usuario ha pulsado la opción de mostrar el árbol de una oración	
Secuencia normal	**Paso**	**Acción**
	1	El sistema analizará cada palabra con las demás de la oración para saber las dependencias halladas
	2	El sistema dibujará el árbol a partir de las dependencias encontradas entre las palabras
Postcondición	El usuario podrá ver el árbol de la oración propuesta	
Excepción	**Paso**	**Acción**
	-	-
Importancia	Vital	
Estabilidad	Alta	

Figura 24: Funcionalidad 20 - Mostrar árbol

UC-0021	Ver información	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber completado **UC-0020**	
Descripción	El usuario podrá observar la información morfológica de cada nodo del árbol	
Precondición	El usuario quiere visualizar la información de una palabra de la oración	
Secuencia normal	Paso	Acción
	1	El usuario pondrá el ratón encima de un nodo del árbol.
	2	El sistema reconocerá el nodo y cargará la información de esa palabra
	3	El sistema mostrará la información de esa palabra al usuario
Postcondición	El usuario podrá ver la información de la palabra de forma dinámica	
Excepción	Paso	Acción
	-	-
Importancia	Vital	
Estabilidad	Alta	

Figura 25: Funcionalidad 21 - Ver información

UC-0022	Confirmar anotación	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber completado **UC-0015**	
Descripción	El usuario podrá guardar la información del patrón encontrado en el corpus cargado	
Precondición	El usuario ha aplicado un patrón, lo ha encontrado en el corpus y quiere guardar esa información	
Secuencia normal	Paso	Acción
	1	El usuario pulsará la opción de guardar cambios en el corpus
	2	El sistema guardará la información producida de la búsqueda del patrón en el corpus cargado
Postcondición	El usuario podrá ver en su corpus como se ha actualizado la información por el patrón utilizado.	
Excepción	Paso	Acción
	1	Si no se ha encontrado ningún patrón y el usuario pulsa esta opción no se actualizará nada. Además, se notificará al usuario de que no se ha podido actualizar.
Importancia	Vital	
Estabilidad	Alta	

Figura 26: Funcionalidad 22 - Confirmar anotación

UC-0023	Deshacer anotación	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	El usuario debe haber completado **UC-0022**	
Descripción	El usuario podrá deshacer la anotación guardada anteriormente	
Precondición	El usuario quiere volver a tener la información en el corpus como al principio	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsará la opción de deshacer la anotación realizada anteriormente.
	2	El sistema eliminará los cambios anteriores, y volverá a un corpus inicial
Postcondición	El usuario podrá ver en su corpus como tiene la información como inicialmente	
Excepción	**Paso**	**Acción**
	1	Si el usuario no ha realizado ninguna actualización de anotación en el corpus, se notificará al usuario de que el corpus no ha sido modificado y se presenta como al inicio
Importancia	Vital	
Estabilidad	Alta	

Figura 27: Funcionalidad 23 - Deshacer anotación

4.3.2.2 Casos de uso del sistema

UC-0024	Encontrar patrones	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	Ninguna	
Descripción	El sistema encontrará el/los patrón/patrones que haya escrito el usuario en el corpus cargado	
Precondición	El sistema tiene almacenado el patrón y el corpus	
Secuencia normal	**Paso**	**Acción**
	1	El sistema deberá ejecutar la herramienta Grew, para poder encontrar las oraciones que cumplen el patrón propuesto
	2	El sistema devuelve las oraciones en las que se cumple dicha propiedad
Postcondición	El sistema tiene almacenadas las oraciones que cumplen el patrón de búsqueda	
Excepción	**Paso**	**Acción**
	1	Si no se ha encontrado ningún patrón en el corpus cargado, se notificará del error.
Importancia	Vital	
Estabilidad	Alta	

Figura 28: Funcionalidad 24 - Encontrar patrones

UC-0025	Aplicas reglas	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	Ninguna	
Descripción	El sistema aplicará las reglas que tenga cargadas un usuario a una oración, con el fin de simplificar la oración	
Precondición	El usuario dispone de reglas cargadas y ha seleccionado una oración	
Secuencia normal	**Paso**	**Acción**
	1	El sistema buscará que regla aplicar de todas las que tiene cargadas el usuario. Dependiendo de cuál se ajusta mejor a la oración seleccionada
	2	El sistema hará las operaciones necesarias con la regla elegida en la oración
Postcondición	El sistema devolverá la oración simplificada por la regla	
Excepción	**Paso**	**Acción**
	1	No había ninguna regla que pudiera reducir la oración. Se notificará al usuario
	2	El usuario no había definido reglas, por lo que no se ha podido aplicar ninguna. Se notificará al usuario.
Importancia	Vital	
Estabilidad	Alta	

Figura 29: Funcionalidad 25 - Aplicar reglas

UC-0026	Analizar dependencias	
Versión	1.0	
Autor	Carlos Gavidia Ortiz	
Fuentes	Cliente	
Dependencias	Ninguna	
Descripción	El sistema analizará que palabras dependen de otras en una oración y cuál es la dependencia que la unen	
Precondición	El usuario ha cargado un corpus y ha seleccionado las etiquetas verbales	
Secuencia normal	**Paso**	**Acción**
	1	El sistema analizará cada oración del corpus
	2	Por cada oración del corpus, se analizará como se disponen las palabras en ella y que palabras dependen unas de otras.
	3	Se almacenará esa información de las dependencias de las palabras.
Postcondición	Se ha analizado todas las palabras del corpus y se ha almacenado la información de las dependencias de todas ellas	
Excepción	**Paso**	**Acción**
	1	No hay ninguna oración cargada en el sistema, por lo que no se puede analizar ninguna palabra. Se notificará al usuario
Importancia	Vital	
Estabilidad	Alta	

Figura 30: Funcionalidad 26 - Analizar dependencias

4.4. Arquitectura de la aplicación

El desarrollo del proyecto ha sido implementado mediante una arquitectura cliente-servidor [33], siendo el cliente, la aplicación web que interactuará con el usuario y mostrará los resultados. Este cliente hará peticiones al servidor, donde se realizarán las llamadas con la herramienta Grew [11], para la búsqueda de patrones y la aplicación de reglas en un corpus lingüístico.

A continuación se puede apreciar en la Figura 31, un esquema de como se conectan todos los módulos y tecnologías empleadas, que se disponen en el proyecto.

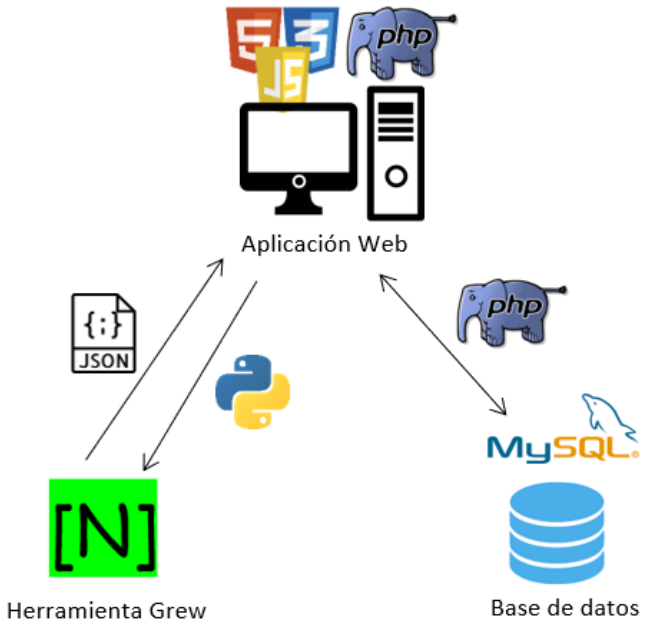


Figura 31: Visión general del proyecto

La estructura del proyecto se divide en tres bloques principales:

1. Por un lado en la aplicación web, será donde los usuarios tengan una interfaz, para realizar todas las funciones en sus textos y visualizar los resultados.

2. La aplicación web almacenará todo el contenido de la herramienta en una base de datos relacional, se utiliza MySQL para ello.

3. Para realizar las operaciones de búsqueda de patrones, y aplicación de reglas a las oraciones, la aplicación consultará con la herramienta Grew mediante python. Una vez se realicen las operaciones, los resultados serán devueltos en formato JSON para que se traten en la aplicación y se muestren los resultados al usuario.

[11] Herramienta desarrollada en la sección 1.3

4.5. Modelo de datos

En esta sección se describe el modelo Entidad-Relación y a continuación su implementación mediante una base de datos relacional.

4.5.1. Modelo Entidad-Relación

En la Figura 32, se muestra el diagrama entidad relación del sistema, donde se almacenará la información de los usuarios, las reglas creadas por los usuarios y los documentos guardados. Cada usuario puede guardar o crear N textos y N reglas que estarán asociados al usuario por el id del usuario.

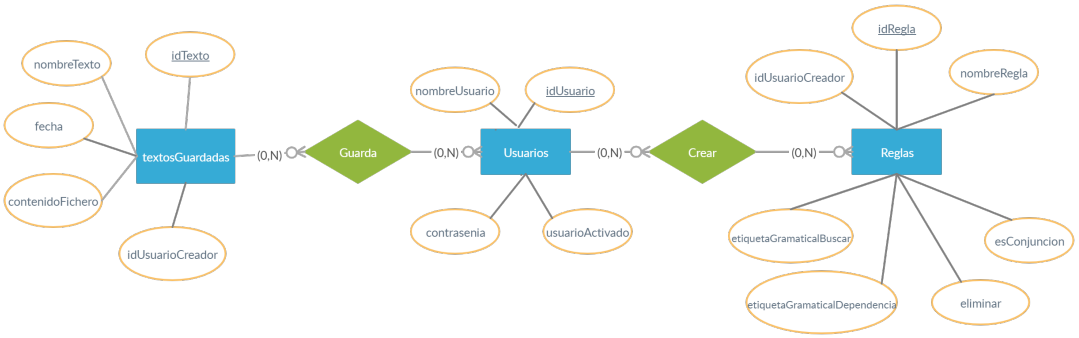


Figura 32: Modelo entidad-relación

Se ha optado por la implementación de una base con datos relacional con el fin de garantizar el almacenamiento de los datos y la posibilidad de que puedan ser escalables.

4.5.2. Base de datos relacional

El modelo Entidad-Relación se ha implementado mediante una base de datos relacional de tipo MySQL. En la Figura 33, se muestran las tablas y las relaciones entre las tablas.

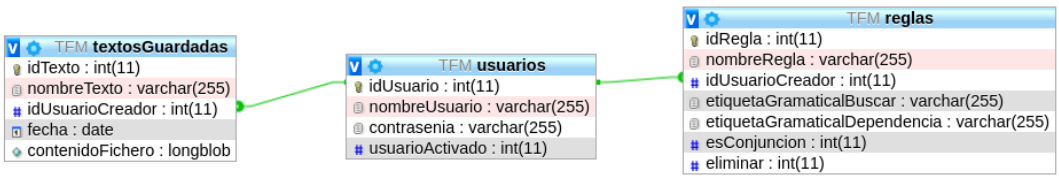


Figura 33: Esquema de la base de datos del sistema

A continuación se describen cada una de las tablas:

Usuarios

Se trata de la tabla donde se guardarán todos los usuarios que se registren en la aplicación. Las columnas de la tabla son:

- **IdUsuario**: es la clave primaria de la tabla, se utiliza para identificar a cada usuario. El tipo de este campo, será entero e incremental por cada inserción de un nuevo usuario en la base de datos.

- **nombreUsuario**: es un campo de tipo varchar de longitud máxima de 255 caracteres. Se trata del nombre con el que el usuario se ha dado de alta en la aplicación.
- **contrasenia**: al igual que el campo anterior, es un campo de tipo varchar de longitud máxima de 255 caracteres. Se almacena la contraseña hasheada del usuario.
- **usuarioActivado**: se trata de un campo de tipo entero, que almacenará 1 si el usuario ha activado su usuario y ha entrado en la aplicación o 0 en caso contrario.

Reglas

Es la tabla que almacenará todas las reglas que los usuarios hayan creado en la aplicación. Dicha tabla se unirá con la tabla usuarios mediante el id del usuario, sus campos son:

- **idRegla**: se trata de la clave primaria de la tabla, es de tipo entero e incremental por cada inserción de una nueva regla. Con este campo se identificará cada regla.
- **nombreRegla**: es un campo de tipo varchar con longitud máxima de 255 caracteres. Identifica el nombre, que haya escrito el usuario a la regla.
- **idUsuarioCreador**: es un campo de tipo entero, será clave foránea de la tabla usuarios para identificar el id del usuario que ha creado la regla. Además será una clave primaria, con el fin de que haya reglas iguales pero de diferentes usuarios.
- **etiquetaGramaticalBuscar**: es un campo de tipo varchar de longitud máxima 255 caracteres. Corresponde a uno de los campos a completar cuando el usuario cree una regla. Se espera que el usuario escriba la etiqueta gramatical de la palabra de la oración que desea buscar.
- **etiquetaGramaticalDependencia**: se trata de un campo de tipo varchar de longitud máxima de 255 caracteres. Corresponde a otro campo en el formulario que deberá completar el usuario al crear una regla. En ella, deberá determinar la etiqueta gramatical de la dependencia que se buscará.
- **esConjunción**: es un campo entero, corresponde a otro campo del formulario para la creación de una regla. Será evaluada a 1 si la palabra que se desea buscar es una conjunción o 0 en caso contrario.
- **eliminar**: es un campo entero, y corresponde al último campo del formulario para la creación de reglas. Podrá tener el valor 1, si una vez encontrada la palabra buscada, se desea eliminar dicha palabra y sus dependientes, o podrá tener valor 0 si no se desea esa eliminación.

TextosGuardados

Es una tabla utilizada para guardar los textos que el usuario tenga interés en almacenar. Esta tabla se unirá con la tabla de usuarios mediante el id de usuarios. Sus campos son:

- **idTexto**: es un campo entero, e incremental por cada inserción nueva, corresponde a la clave primaria de la tabla. Con dicho campo se podrá identificar a un texto guardado.
- **nombreTexto**: se trata de un campo varchar de longitud máxima de 255 caracteres. Este campo corresponde al nombre que haya indicado el usuario que desea guardar el texto.
- **idUsuarioCreador**: es un campo entero, será clave foránea de la tabla usuarios para identificar al usuario que ha guardado el texto. Además será una clave primaria, con el fin de que haya textos con el mismo nombre pero de diferentes usuarios.
- **fecha**: se trata de un campo de tipo fecha, con él se podrá determinar la fecha en la que el usuario guardó el texto.

- **contenidoFichero**: es un campo de tipo longblob, dicho tipo es utilizado para almacenar ficheros de mucho tamaño. Su función será la de almacenar todo el texto y la información morfológica de cada palabra del texto, que quiera guardar el usuario.

Capítulo 5: Diseño de la aplicación

En este capítulo se describirán todas las funcionalidades implementadas en el proyecto y los recursos necesarios para su desarrollo, como clases, scripts, ficheros...

5.2. Registrar usuario

Los ficheros encargados de dar de alta a un usuario, se encuentran en la carpeta *registrar*, y son *registrar.php* e *index.php*.

El fichero *index.php*, se encargará de mostrar la interfaz al usuario, cargará los ficheros de estilo, e implementará las clases de Boostrap para mostrar tres componentes editables. En el primero el usuario, deberá indicar el nombre y en los dos siguientes la contraseña. Además se ha implementado un botón que enviará la información al fichero *registrar.php*.

El archivo *registrar.php*, se encargará de la parte lógica. Recibirá los datos desde *index.php*, mediante un método HTTP POST. A continuación comprobará en la base de datos, que no existe ningún usuario con ese nombre, y que las contraseñas del usuario coinciden entre ellas. En caso afirmativo, se procede a insertar en la base de datos, ese usuario con esa contraseña hasheada mediante la función *"passwordhash"*. Si no cumple los requisitos, no se dará de alta y se notificará en la interfaz al usuario del error.

En la Figura 34, se puede observar el fragmento de código utilizado, para registrar a un usuario.

```php
if($mysqli)
{
    //Recojo los valores del registro
    $usuario=$_POST['usuario'];
    $password=$_POST['password'];
    $password2=$_POST['password2'];


    //if ($usuario!="" && $password!="" && $password2!="" ){

        if ($password!=$password2 ){
            $notificacion .="Las contraseñas no coinciden";
        }
        else{
            $sql="select * from usuarios where email='$usuario'";

            $resultado = mysqli_query($mysqli, $sql);
            $numregistros=$resultado->num_rows;
            $obj = $resultado->fetch_object();
            //si existe ese usuario

            if ($numregistros==0){

                $contrasena = password_hash($password, PASSWORD_DEFAULT);
                $query = "INSERT INTO usuarios VALUES ('$usuario', '$contrasena')";
                $resultado = mysqli_query($mysqli, $query);
                //se crea la sesion y se guarda las variables de usuario y contraseña
                $_SESSION['usuario']=$usuario;
                $_SESSION['password']=$contrasena;
                header('Location:' . ROOT_PATH);
                exit();

            }
            else{
                $notificacion .="Ya existe ese usuario creado";
            }

        }
```

Figura 34: Registro de usuarios

5.3. Inicio de sesión

Los ficheros de la aplicación encargados de realizar el inicio de la sesión de un usuario, están presentes en la carpeta *login*, y son *login.php* e *index.php*.

El fichero *index.php*, estará dedicado a mostrar la interfaz al usuario. Como se puede apreciar, al igual que en el registro, se hace uso de clases Bootstrap para conseguir una interfaz basada en dos componentes y un botón. Enviando los datos mediante un protocolo HTTP POST al archivo *login.php*.

El archivo *login.php*, espera recibir los datos llegados desde *index.php*, además se hará uso de la función "*tratarTexto*", utilizada para eliminar todo rastro de caracteres extraños que puedan llegar a incluirse en el envío de los datos. Una vez se reciban y se traten, se conectará con la base de datos y se comprobará si existe el usuario. En caso afirmativo se comprobará que la contraseña está relacionada con ese usuario, para ello se usará la función: "*passwordverify*", ya que la contraseña está hasheada en la base de datos. Finalmente si tanto usuario como contraseña existen, se creará una variable de sesión para almacenar el nombre del usuario. En caso contrario se notificará al usuario del campo incorrecto.

A continuación en la Figura 35, se puede observar el fragmento de código utilizado, para comprobar que el usuario existe en la aplicación.

```php
session_start();
$mysqli = new mysqli("localhost", "root", "", "filologia");
if (mysqli_connect_errno()) {
    printf("Error de conexión: %s\n", mysqli_connect_error());
    exit();
}

if($mysqli)
{
    //Recojo los valores del registro
    $usuario=tratarTexto($_POST['usuario']);
    $password=tratarTexto($_POST['password']);

    if ($usuario!="" && $password!="" ){


    $sql="select * from usuarios where email='$usuario'";

    $resultado = mysqli_query($mysqli, $sql);
    $numregistros=$resultado->num_rows;
    $obj = $resultado->fetch_object();
    //si existe ese usuario

    if ($numregistros!=0 and password_verify($password, $obj->contrasenia)){

        //se crea la sesion y se guarda las variables de usuario y contraseña
        $_SESSION['usuario']=$usuario;
        $_SESSION['password']=$password;
        header('Location:' . ROOT_PATH);
        exit();

    }
    else{
        $notificacion .="Usuario y/o contraseña incorrectos";
    }

    }
```

Figura 35: Inicio de sesión de usuarios

5.4. Lectura de corpus lingüísticos

En esta sección se detallan los ficheros utilizados para la lectura del corpus lingüístico, para que sea analizado. El fichero principal es *index.php*, que está en el la raíz del proyecto. Será el encargado de mostrar la interfaz al usuario para que seleccione un archivo local, de un corpus lingüístico que quiera analizar. A continuación, redirigirá al usuario, al fichero *upload.php*. Su función será la de cargar archivos con extensión ***.conll***. Se puede observar en la Figura 36, el fragmento de código para la carga de archivos.

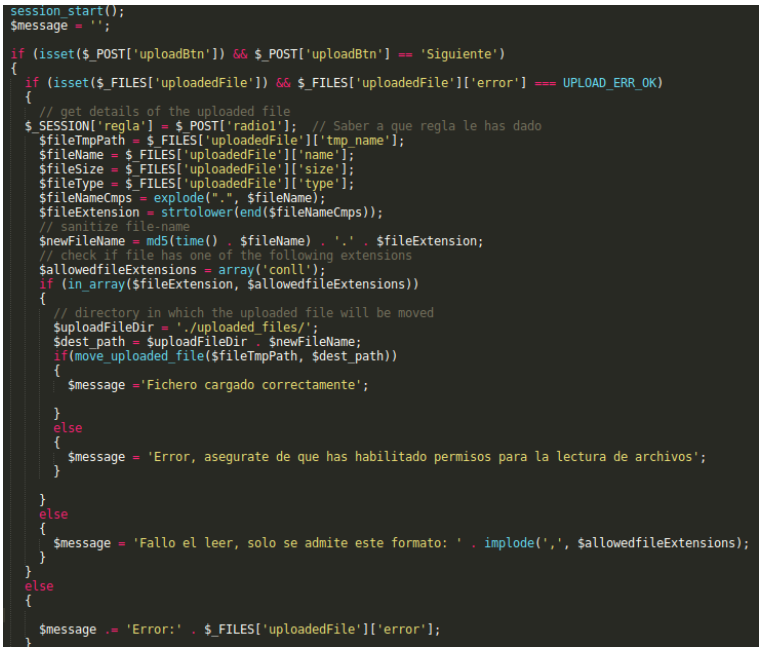


Figura 36: Cargar archivo con formato .conll

Una vez cargado el fichero, se comprobará que el contenido del archivo subido cumple con el formato ***CoNLL***, para ello se utilizará el archivo *index.php* de la carpeta *comprobarArchivo*, que leerá y comprobará que cada línea, presenta 10 columnas separadas por un espacio o un tabulado y que cada columna corresponde a los campos: id, forma, lema UPOS, XPOS, Características, Cabecera, Deprel, Deps, Miscelánea.

En caso de que las columnas del archivo no cumplan con el formato anterior, ya que hay más o menos, o que a pesar de tener 10 columnas los campos no se corresponden en el orden mencionado anteriormente, se redirigirá al usuario a al fichero *index.php* de la carpeta *formatoArchivo*. Que mostrará una interfaz al usuario donde pedirá que seleccione, a que columna del fichero corresponden los campos: id, forma, lema UPOS, XPOS, Características, Cabecera, Deprel, Deps, Miscelánea. Y el fichero *formatoArchivo.php*, será el encargado de la lógica. Esperará las columnas que haya seleccionado el usuario y construirá un archivo ***.conll***, de 10 columnas. Reorganizando la información del archivo subido por el usuario. A continuación se puede ver el algoritmo desarrollado para reescribir el fichero con las columnas reorganizadas, en la Figura 37.

Figura 37: Reorganizar las columnas del archivo subido por el usuario

Una vez la herramienta tenga un fichero ***.conll*** correcto, volveremos a *index.php*, donde comenzará el proceso de lectura del archivo. Se leerá todo el contenido del fichero, y se guardarán todas las palabras de cada oración en arrays, la forma de guardarlo, es en un array general de N posiciones, estas N posiciones será en número de oraciones del corpus. Y por cada posición habrá otro array, que tendrá 10 posiciones, en cada una se guardarán los campos del formato CoNLL, y con ello se tendrá toda la información del corpus. Durante la lectura del corpus, también se contabilizará el número de verbos leídos, a partir de la etiqueta gramatical de las palabras, para poder determinar cuántos verbos posee cada oración y así mostrar esa información al usuario en la interfaz. A continuación se puede observar en la Figura 38, el algoritmo para la lectura del contenido y la lectura de de verbos:

```
        array_push($oracionesHead, $head);
        array_push($oracionesDeprel, $deprel);
        array_push($oracionesDepss, $deps);
        array_push($oracionesMisc, $misc);
        $id=array();
        $form=array();
        $lemma =array();
        $cpos=array();
        $pos=array();
        $feat=array();
        $head=array();
        $deprel=array();
        $deps=array();
        $misc=array();
        $oracionesVerbos[$contador] = $verbos;
        $contador=$contador+1;
        if ($verbos>1){
            $oracionesDobleVerbo=$oracionesDobleVerbo+1;
        }
        else{
            $oracionesUnVerbo=$oracionesUnVerbo+1;
        }
        $verbos=0;
    }
    $anterior=$token;
    array_push($id, $token);
    $token = strtok(" \n\t");
    array_push($form, $token);
    $token = strtok(" \n\t");
    array_push($lemma, $token);
    $token = strtok(" \n\t");
    array_push($cpos, $token);
    $token = strtok(" \n\t");
    array_push($pos, $token);
    $listaVerbos=array();
    $listaVerbos = explode(',', $_SESSION['listaVerbos']);
    if (in_array(strtoupper($token), $listaVerbos)){
        $verbos=$verbos+1;
    }
    $token = strtok(" \n\t");
    array_push($feat, $token);
    $token = strtok(" \n\t");
    array_push($head, $token);
    $token = strtok(" \n\t");
    array_push($deprel, $token);
    $token = strtok(" \n\t");
    array_push($deps, $token);
```

Figura 38: Lectura del contenido de un archivo CoNLL

5.5. Seleccionar oraciones del corpus cargado

Para la selección de oraciones, será necesario el cargado del corpus, mencionando en el punto anterior. Terminado ese proceso se mostrarán las oraciones al usuario, para que pueda seleccionar una, todas o ninguna y realizar operaciones con esas oraciones seleccionadas.

El fichero *index.php* y *mostrarOraciones.php* de la carpeta *mostrarOraciones*, serán los encargados de mostrar las oraciones leídas en el corpus al usuario, junto con la opción de tener una selección múltiple para seleccionar una o varias oraciones. A continuación se puede observar en la Figura 39, el proceso para mostrar las oraciones cargadas:

Figura 39: Mostrar oraciones del corpus cargado

Después de la selección de las oraciones, se podrán descargar archivos, aplicar reglas al corpus, mostrar resultados... Diversas funciones que se detallarán en profundidad en las siguientes secciones.

5.6. Creación de patrones y reglas lingüísticos

Esta funcionalidad estará disponible en la vista avanzada de la herramienta, es decir la que presenta 4 partes en la pantalla. El fichero encargado de la creación de los patrones y/o reglas lingüísticos es *index.php*. Se ha implementado una interfaz para el usuario pueda redactar un patrón o una regla en un componente *textfield*, o que pueda abrir un archivo con ese patrón o regla. En cualquiera de las dos opciones se leerá el contenido y se guardará en un archivo temporal, dicho patrón para luego poder ser aplicado. A continuación en la Figura 40, se puede observar el código de la interfaz de esta funcionalidad:

Figura 40: Creación de patrón lingüístico

5.7. Creación y edición de reglas

Esta funcionalidad estará disponible en la vista simple de la herramienta. Los ficheros necesarios para la creación y edición de las reglas son *creadorReglas.php* y *editorReglas.php*, en la carpeta *vistaUsuario*. En el primero de ellos, se presenta al usuario un formulario para que pueda escribir los tipos de regla que quiere crear para aplicar a un corpus. Una vez creado, se insertará en la base de datos las reglas, relacionándolas con al usuario que haya iniciado la sesión. El segundo archivo mencionado será dedicado a la edición de las reglas que tenga el usuario, y para ello se hará una actualización de las reglas en la base de datos del usuario. A continuación se puede observar en la Figura 41 la inserción y actualización de reglas en la base de datos del usuario de la sesión:

Figura 41: Creación y edición de reglas lingüísticas

5.8. Buscar patrón en el corpus cargado

El fichero encargado de realizar la búsqueda del patrón en el corpus cargado, es *index.php*. En la Figura 42, se puede observar que para realizar la búsqueda, se hará una llamada a la herramienta *Grew*, previamente instalada en el sistema. Los parámetros necesarios para ejecutar este comando serán el patrón escrito y el corpus cargado. La ejecución interna que realizará este comando, es buscar de forma automática si el patrón escrito, se cumple en alguna de las oraciones del corpus. Un ejemplo de patrón sencillo a buscar, es "buscar aquellas oraciones en las que se cumple que existe un verbo con un tiempo verbal en pasado que depende del sujeto de la oración". Será el usuario quien determine la complejidad de la búsqueda, y en medida de esta complejidad se reducirán las oraciones en las que se cumple dicha propiedad.

```
if (isset($_POST['grew']) && $_POST['grew'] == 'Aplicar reglas')
{
    if ($_SESSION['relleno']==1){
    $_SESSION['relleno']=0;

    $exit = shell_exec('rew grep -pattern '.$_SESSION['patron'] '-i '.$_SESSION['corpus'].' 2>&1');

    $myArray = explode(',', $exit);
    error_reporting(0);

    $idOracionesModificadas=array();
    $nodosOracionesModificadas=array();
    $primero=true;
    for ($i=0;$i<count($myArray);$i++)
    {
        //echo $i;
        $myArray2 = explode(':', $myArray[$i]);
        //print_r( $myArray2);
        if (strpos($myArray2[0], 'sent_id') !== false){

            if ($primero!=true){
                array_push($nodosOracionesModificadas,$nodosOraciones );
            }
            else{
                $primero=false;
            }
            $palabra=str_replace('"','',$myArray2[1]);
            array_push($idOracionesModificadas,$palabra);
            $nodosOraciones=array();

        }
        if (strpos($myArray2[0], 'matching') !== false){

            $palabra=str_replace('"','',$myArray2[3]);
            array_push($nodosOraciones,$palabra);

        }
        else if (strpos($myArray2[0], 'sent_id') == false && strpos($myArray2[0], 'edges') == false && strpos($myArray2[0], 'label') === false && strpos(
            $myArray2[0], 'target') === false
            && strpos($myArray2[0], 'e1') === false){

            $palabra=str_replace('"','',$myArray2[1]);
            array_push($nodosOraciones,$palabra );
        }
```

Figura 42: Búsqueda de patrón lingüístico en el corpus cargado

Las oraciones que se cumplan estas propiedades descritas, serán devueltas en formato JSON, como se puede apreciar en la Figura 43, donde se puede observar un ejemplo de oraciones devueltas tras la aplicación de un patrón. En la imagen se puede observar tanto el identificador de la oración, los nodos donde se cumple la propiedad y la dependencia por la que se unen esos nodos. En las siguientes secciones se desarrollarán dichas funcionalidades, con los resultados devueltos.


```
[
  {
    "sent_id": "Europar.550_00496",
    "matching": {
      "nodes": { "V": "16", "DE": "19", "A": "14" },
      "edges": {
        "e2": { "source": "16", "label": "de_obj", "target": "19" },
        "e1": { "source": "16", "label": "a_obj", "target": "14" }
      }
    }
  },
  {
    "sent_id": "emea-fr-test_00478",
    "matching": {
      "nodes": { "V": "33", "DE": "32", "A": "35" },
      "edges": {
        "e2": { "source": "33", "label": "de_obj", "target": "32" },
        "e1": { "source": "33", "label": "a_obj", "target": "35" }
      }
    }
  },
```


Figura 43: Resultados devueltos por Grew, para comprobar un patrón en un corpus

5.9. Reducción de oración por una regla lingüística

Esta funcionalidad solo estará presente en la vista avanzada de la herramienta, no en la vista simple. El fichero inicial para realizar la reducción de una oración a partir de una regla, comienza en *index.php*. Como se puede ver en la Figura 44, se ejecutará un archivo python en la línea de comandos, y realizará la llamada a Grew, con la función *grew.run*, pasando como argumento la oración que se quiere reducir y la regla que va a conseguir dicha reducción.

```
//echo $exit;
//ARRAY CON LOS W, ES DECIR LOS IDS QUE SE QUEDAN
if (strpos($exit, '{')=== false){
    $verbosA=0;

    for ($i=0;$i<count($pos);$i++){
        $listaVerbos=array();
        $listaVerbos = explode(',', $_SESSION['listaVerbos']);
        if (in_array(strtoupper($pos[$i]), $listaVerbos)){
                $verbosA=$verbosA+1;
        }
    }
    if ($verbosA>1){
        if (!isset($_SESSION["oracionesCompuestasAnalizadas"])){
        $_SESSION["oracionesCompuestasAnalizadas"]=array();
            array_push($_SESSION["oracionesCompuestasAnalizadas"], $_SESSION['oracionesSent_id2'][$_SESSION['numOracion']-1]);
        }
        if (!in_array($_SESSION['oracionesSent_id2'][$_SESSION['numOracion']-1],$_SESSION["oracionesCompuestasAnalizadas"])){
            array_push($_SESSION["oracionesCompuestasAnalizadas"], $_SESSION['oracionesSent_id2'][$_SESSION['numOracion']-1]);

        }
    }?>
    <script>
    swal("La regla no ha simplificado la oración", "Comprueba la regla o escribe otra distinta para aplicar a esa oración","error");
    </script>
<?php }
if(strpos($exit, '{')!== false){?>
    <script>
    swal("Se ha aplicado la regla correctamente", "Puedes ver los resultados de aplicar esa regla a estar oración","success");
    </script>
    <?php
$lineas=array();
for ($i=0;$i<strlen($exit);$i++){
    if ($exit[$i]=='W' && $exit[$i+2]=="'"&& $exit[$i+3]==":"&& $exit[$i+4]==" "&& $exit[$i+5]=="["){
        array_push($lineas, $exit[$i+1]);
    }
}
//ARRAY DE ARRAY CON LAS DEPENCIAS DE CADA LINEA
```

```
import grew
grew.init()
f= open("tmp/oracion.txt","r")
contents =f.read()
g = grew.graph(contents)
f= open("tmp/regla.txt","r")
r=f.read()
print(grew.run(r, g, 'passiveAgt'))
```

Figura 44: Tratamiento de los resultados devueltos tras aplicar una regla a una oración

Un ejemplo de oración y regla que espera recibir Grew, es la que aparece en la Figura 45

```
graph{
W1 [phon="Cette",cat=D];
W2 [phon="exposition",cat=N];
W3 [phon="nous",cat=CL];
W4 [phon="apprend",cat=V, m=pastp];
W5 [phon="que",cat=C];
W6 [phon="dès",cat=P];
W7 [phon="le",cat=D];
W8 [phon="XIIe",cat=A];
W9 [phon="siècle",cat=N];
W10 [phon=",",cat=PONCT];
W11 [phon="à",cat=P];
W12 [phon="Dammarie-sur-Saulx",cat=N];
W13 [phon=",",cat=PONCT];
W14 [phon="entre",cat=P];
W15 [phon="autres",cat=A];
W16 [phon="sites",cat=N];
W17 [phon=",",cat=PONCT];
W18 [phon="une",cat=D];
W19 [phon="industrie",cat=N];
W20 [phon="métallurgique",cat=A];
W21 [phon="existait",cat=V];
W22 [phon=".",cat=PONCT];
W2 -[det]->W1;
W4 -[suj]->W2;
W4 -[a_obj]->W3;
W4 -[obj]->W5;
W21 -[mod]->W6;
W9 -[det]->W7;
W9 -[mod]->W8;
W6 -[obj.p]->W9;
W21 -[ponct]->W10;
W21 -[mod]->W11;
W11 -[obj.p]->W12;
W21 -[ponct]->W13;
W21 -[mod]->W14;
W16 -[mod]->W15;
W14 -[obj.p]->W16;
W21 -[ponct]->W17;
W19 -[det]->W18;
W21 -[suj]->W19;
W19 -[mod]->W20;
W5 -[obj.cpl]->W21;
W4 -[ponct]->W22;
}
```

```
rule passiveAgt {
  pattern {
    V [cat=V, m=pastp];
    V -[aux.pass]-> AUX;
    e: V -[suj]-> SUJ;
    P [phon=par]; V -[p_obj.agt]-> P;
    P -[obj.p]-> A;
  } commands {
    del_node P;
    del_node AUX;
    add_edge V -[suj]-> A;
    add_edge V -[obj]-> SUJ;
  } }
```

Figura 45: Ejemplo de oración y regla

Finalmente se devolverán las palabras, que se mantendrían en la oración, y las nuevas dependencias entre las palabras en formato JSON, como se observa en la Figura 46. Con dichos resultados, se mostrará la nueva oración reducida y se dibujará el nuevo árbol de dependencias de la oración.

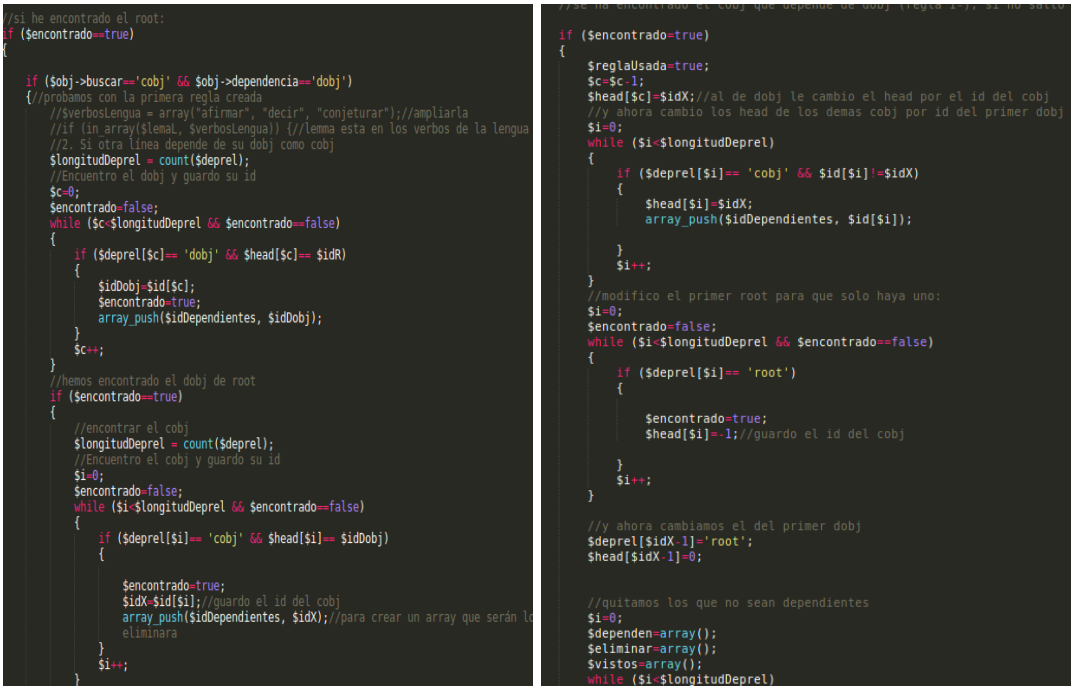
```
[{'W5': ('cat="D", phon="le"', []), 'W6': ('cat="NP", word="chie
n"', [('det', 'W5')]), 'W3': ('cat="V", m="pastp", phon="mord
u"', [('obj', 'W1'), ('suj', 'W6')]), 'W1': ('cat="NP", phon="Jo
hn"', [])}]
```

Figura 46: Resultados de llamada a grew.run

5.10. Utilización de un formulario para creación de reglas

El fichero encargado de aplicar las reglas que haya creado el usuario es *index.php*, en la carpeta *vistaUsuario*. La utilización de las reglas que haya creado el usuario, estará disponible en la vista de simple, no en la avanzada. El usuario deberá de haber creado algunas reglas para que se pueden aplicar a un corpus cargado. Un ejemplo de una regla guardada podría servir para .[el]iminar la subordinada del complemento directo de una oración".

La forma que determinará que una regla es la idónea para aplicar al texto propuesto, será la de recorrer cada una de ellas, e ir comprobando si aplicando una en concreto se consigue reducir el texto. En la Figura 47, se puede observar la lectura de una de ellas y se la comprobación de si es óptimo aplicarla, porque reduce el texto propuesto, o si no es así, seguirá con la siguiente y así sucesivamente hasta encontrar la que consiga dicho objetivo.



```
//si he encontrado el root:
if ($encontrado==true)
{

   if ($obj->buscar=='cobj' && $obj->dependencia=='dobj')
   {//probamos con la primera regla creada
       //$verbosLengua = array("afirmar", "decir", "conjeturar");//ampliarla
       //if (in_array($lemaL, $verbosLengua)) {//lemma esta en los verbos de la lengua
       //2. Si otra línea depende de su dobj como cobj
       $longitudDeprel = count($deprel);
       //Encuentro el dobj y guardo su id
       $c=0;
       $encontrado=false;
       while ($c<$longitudDeprel && $encontrado==false)
       {
           if ($deprel[$c]== 'dobj' && $head[$c]== $idR)
           {
               $idDobj=$id[$c];
               $encontrado=true;
               array_push($idDependientes, $idDobj);
           }
           $c++;
       }
       //hemos encontrado el dobj de root
       if ($encontrado==true)
       {
           //encontrar el cobj
           $longitudDeprel = count($deprel);
           //Encuentro el cobj y guardo su id
           $i=0;
           $encontrado=false;
           while ($i<$longitudDeprel && $encontrado==false)
           {
               if ($deprel[$i]== 'cobj' && $head[$i]== $idDobj)
               {

                   $encontrado=true;
                   $idX=$id[$i];//guardo el id del cobj
                   array_push($idDependientes, $idX);//para crear un array que serán lo
                   eliminara
               }
               $i++;
           }
```

```
if ($encontrado=true)
{
    $reglaUsada=true;
    $c=$c-1;
    $head[$c]=$idX;//al de dobj le cambio el head por el id del cobj
    //y ahora cambio los head de los demas cobj por id del primer dobj
    $i=0;
    while ($i<$longitudDeprel)
    {
        if ($deprel[$i]== 'cobj' && $id[$i]!=$idX)
        {
            $head[$i]=$idX;
            array_push($idDependientes, $id[$i]);

        }
        $i++;
    }
    //modifico el primer root para que solo haya uno:
    $i=0;
    $encontrado=false;
    while ($i<$longitudDeprel && $encontrado==false)
    {
        if ($deprel[$i]== 'root')
        {

            $encontrado=true;
            $head[$i]=-1;//guardo el id del cobj

        }
        $i++;
    }

    //y ahora cambiamos el del primer dobj
    $deprel[$idX-1]='root';
    $head[$idX-1]=0;


    //quitamos los que no sean dependientes
    $i=0;
    $dependen=array();
    $eliminar=array();
    $vistos=array();
    while ($i<$longitudDeprel)
```

Figura 47: Aplicación de una regla a un texto

Actualmente el proyecto está desarrollado para que los usuarios puedan crear varios tipos de reglas. Para poder añadir otros tipos de reglas habría que ampliar esta sección con la implementación necesaria de las reglas a añadir.

5.11. Mostrar árbol de dependencias de los textos

El fichero encargado de pintar el árbol de dependencias, es *index.php*. Para ello, primero se ha realizado un algoritmo para encontrar la raíz de la oración y recursivamente encontrar cada hijo y los hijos de los mismos. A medida que se realizaba ese proceso se concatenaban el nodo padre con los nodos hijos en una cadena de caracteres, como se puede observar en la Figura 48.

```php
$arbol ='var config = {
    container: "#OrganiseChart-simple"
};';
//padre
$padre='var parent_node = {
    text: { name:"'.$formR.'"}
};';
$idRaiz=$idR;
$raiz="parent_node";
$i=0;
$iA=-1;
$nombre="child_";
$n=1;
$array1=array();
$array2=array();
$array3=array();
$array4=array();
$longitud = count($id2);
while ( $longitud!=0 ){
    $longitudArray=count($id2);
    if ($i==$longitudArray){
        $i=0;
        $iA=$iA+1;
        $longitudA=count($array1);
        if ($iA==$longitudA){
            break;
        }
        $idRaiz=$array1[$iA];
        $raiz=$nombre;
        $raiz=$raiz.$array3[$iA];
    }
    if ($head2[$i]==$idRaiz){
        array_push($array1,$id2[$i]);
        array_push($array2,$raiz);
        array_push($array3,$n);
        if ($form2[$i]=='&amp;quot;'){
            array_push($array4,'(comillas)');
        }
        else{
            array_push($array4,$form2[$i]);
        }
        //elimino de la lista:
        unset($id2[$i]);
        $id2 = array_values($id2);
        unset($deprel2[$i]);
        $deprel2 = array_values($deprel2);

        $longitud=$longitud-1;
        $n=$n+1;


    }
    else{

        $i=$i+1;
    }

}

$longitud = count($array1);
$hijos='';
$i=0;
while ( $longitud!=0){

    $hijos=$hijos.'var child_'.$array3[$i].' = {
                parent: '.$array2[$i].',
                text: { name: "'.$array4[$i].'" }
            };';

    $i=$i+1;
    $longitud=$longitud-1;
}

//final:
$final= 'var simple_chart_config = [config, parent_node';
$longitud = count($array2);
$i=0;
while ($i<$longitud)
    {

        $final = $final.', child_'.$array3[$i];


        $i=$i+1;
    }
$final = $final.'];';
//todo junto:

$texto=$arbol.$padre.$hijos.$final;
```

Figura 48: Implementación para obtener las palabras de la oración y sus dependientes

A continuación, para mostrar el árbol y la información morfológica de cada nodo, se utiliza una función en Javacript que espera recibir esa cadena mencionada anteriormente, para pintar el árbol. También se utilizaron funciones Dom para añadir un componente *"tooltip"* a cada nodo, para que se activase cuando detectase que el ratón estaba encima de un nodo, mostrando así la información de esa palabra. En la Figura 49 se puede observar dicha implementación:

```php
<div class="chart" id="OrganiseChart-simple">
</div>
<script src="archivosArbol/vendor/raphael.js"></script>
<script src="archivosArbol/Treant.js"></script>

<script>
<?php echo $texto; ?>
</script>


<script>
//mostramos el arbol
    new Treant( simple_chart_config );

//Con esto hacemos que cuando el usuario pase el raton por un nodo, le muestre toda la info del nodo
//necesito esto para poder pasar toda la informacion de cada linea de text a javascript, y asi poder mostrar en el nodo
var form2 = <?php echo json_encode($form1); ?>;
var lemma2 = <?php echo json_encode($lemma1); ?>;
var pos2 = <?php echo json_encode($pos1); ?>;
var feat2 = <?php echo json_encode($feat1); ?>;
var deprel2 = <?php echo json_encode($deprel1); ?>;
    $(document).ready(function() {
        $("p.node-name").each(function() {
            var palabra2 = $(this).text();
            if (palabra2 =='(comillas)'){
                var indice2 = form2.indexOf('&amp;quot;');
            }
            else{
                var indice2 = form2.indexOf(palabra2);
            }// cogemos el indice de esa palabra y la buscamos en los arrays la info:
            var lemmaI2= lemma2[indice2];
            var posI2= pos2[indice2];
            var featD2= feat2[indice2];
            var featI2=featD2.replace(/\|/g, "</br>");//para que el popup quepa todo
            var deprelI2= deprel2[indice2];
            $(this).replaceWith('<div class="tooltip node-name">'+ palabra2 +'<div class="tooltiptext">'+ lemmaI2 +'</br>'+ posI2+'</br>'+featI2+'</br>'+
                deprelI2 +'</br>'+ '</div>');

        });
    });


    function myFunction() {
    document.getElementById("demo").innerHTML = document.getElementById("myP").innerHTML;
    }
```

Figura 49: Desarrollo de la visualización del árbol con Javascript

5.12. Guardado de anotaciones en los textos cargados

El fichero encargado de esta funcionalidad es *index.php*. Una vez se haya aplicado una regla en un corpus lingüístico y se haya conseguido reducir el número de palabras que forman las oraciones, se guardarán las oraciones resultantes en el archivo cargado por el usuario. Y se escribirán en el ultimo campo del fichero (el campo tiene el nombre de miscelánea), "Yes", en aquellas palabras que no se hayan eliminado en el proceso de simplificación, como se puede observar en la Figura 50. Consiguiendo con ello que el usuario tenga una marca o anotación en su corpus, que palabras se ha aplicado la propiedad.

```
$file = fopen($resultado,'w');
$almo='# sent_id = ';
for ($i=0;$i<count($_SESSION['oracionesSent_id2']);$i++){
    fwrite($file,$almo.$_SESSION['oracionesSent_id2'][$i]);
    fwrite($file,"\n");
    for ($j=0;$j<count($_SESSION['oracionesId2'][$i]);$j++){
        fwrite($file,$_SESSION['oracionesId2'][$i][$j]);
        fwrite($file,"	");
        fwrite($file,$_SESSION['oracionesForm2'][$i][$j]);
        fwrite($file,"	");
        fwrite($file,$_SESSION['oracionesLemma2'][$i][$j]);
        fwrite($file,"	");
        fwrite($file,$_SESSION['oracionesCpos2'][$i][$j]);
        fwrite($file,"	");
        fwrite($file,$_SESSION['oracionesPos2'][$i][$j]);
        fwrite($file,"	");
        fwrite($file,$_SESSION['oracionesFeat2'][$i][$j]);
        fwrite($file,"	");
        fwrite($file,$_SESSION['oracionesHead2'][$i][$j]);
        fwrite($file,"	");
        fwrite($file,$_SESSION['oracionesDeprel2'][$i][$j]);
        //escribo o yes o nada depende si se ha aplicado el patron ahi o no:
        if (in_array($_SESSION['oracionesSent_id2'][$i],$_SESSION["idOracionesModificadas"])){
            $encont=false;
            $numero=0;
            while($encont==false){
                if($_SESSION['oracionesSent_id2'][$i]==$_SESSION["idOracionesModificadas"][$numero]){
                    $encont=true;
                }
                else{
                    $numero=$numero+1;
                }
            }
            if (in_array($_SESSION['oracionesId2'][$i][$j],$arrw)){
                fwrite($file,"yes");
            }
            else{
            fwrite($file,"_");
            }

        }
        else
            fwrite($file,"_");

    fwrite($file,"\n");
```

Figura 50: Función para guardar las anotaciones en el corpus cargado

5.13. Cerrar sesión

El archivo *cerrarSesion.php*, se ejecutará cuando el usuario quiera salirse de su sesión y cerrar la sesión actual. Para ello como se puede observar en la Figura 51, además de cerrar la sesión, se destruirá y con ello todas las variables utilizadas y se redirigirá a la página principal.

```php
<?php

    //Sesiones
    session_start();
    // remove all session variables
    session_unset();
    // destroy the session
    session_destroy();
    header("Location: index.php");

?>
```

Figura 51: Cerrar la sesión del usuario

Capítulo 6: Evaluación de usuarios y pruebas

6.2. Metodología

El motivo de la realización de este proyecto, era ayudar a los usuarios que estuvieran habituados a realizar un análisis en el texto y operaciones sobre él, como búsqueda de patrones, creación y aplicación de reglas en las oraciones. Una vez finalizado el proyecto, se instaló en un servidor, para que pudiera probarse por antiguos alumnos de la Facultad de Filología de la Universidad Complutense de Madrid, mediante una ruta web. En la evaluación de la herramienta participaron 3 alumnos voluntarios y una directora del proyecto. Con motivo de guiar a estos usuarios, se preparó una serie de vídeos tutoriales sobre las funcionalidades de la aplicación.

Finalmente tras probar la aplicación, se elaboró un primer formulario con unas cuestiones de respuesta abierta sobre la opinión de la herramienta y posibles mejoras. Y un segundo formulario para valorar el uso de la aplicación en diferentes puntos, con respuestas valoradas del 0 al 5, siendo 0 la nota más baja y 5 la más alta sobre cada cuestión. En el anexo 11.3.7.4 se muestran las preguntas realizadas en ambos formularios.

6.3. Resultados

A continuación se comentan los resultados obtenidos tras la evaluación de los usuarios:

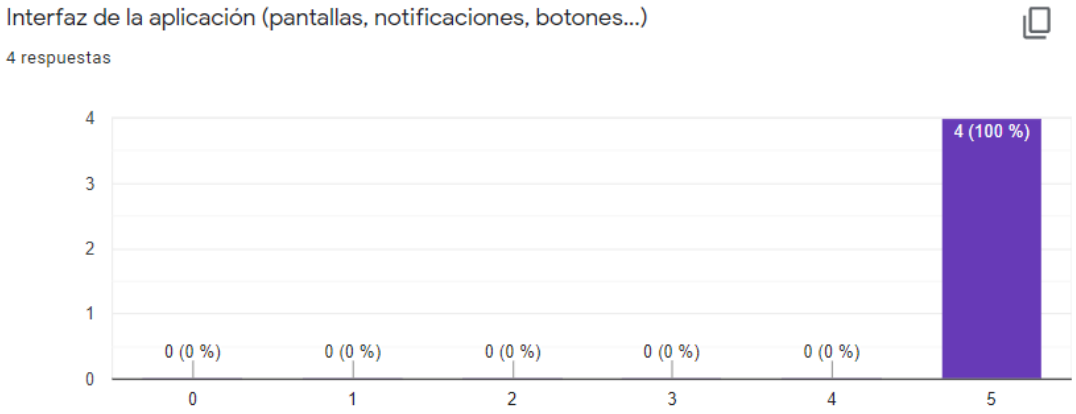


Figura 52: Resultados evaluación con usuarios: Pregunta 1

En la Figura 52, todos los usuarios han valorado con la puntuación máxima que la interfaz de usuario de la herramienta, les ha resultado buena e intuitiva.

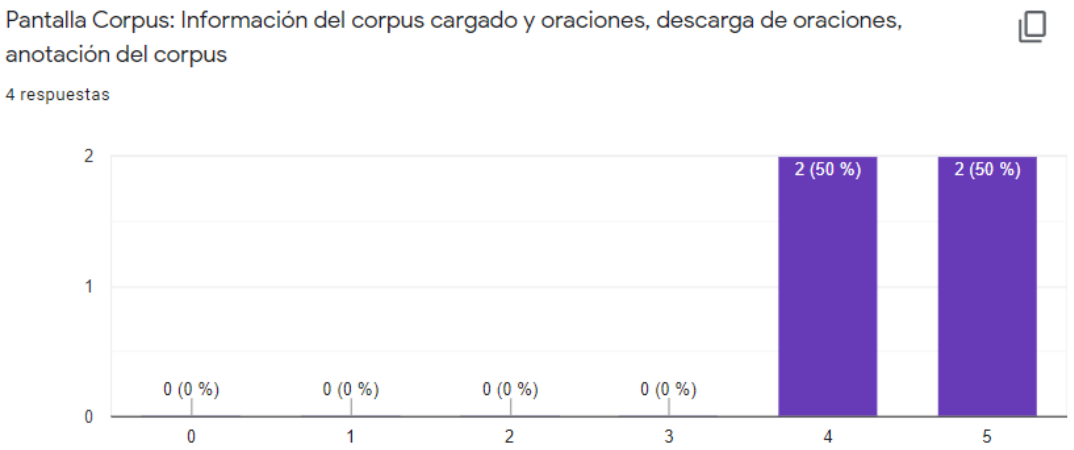


Figura 53: Resultados evaluación con usuarios: Pregunta 2

En la Figura 53, se pregunta por la pantalla de corpus y sus funcionalidades. La mitad de los usuarios han valorado con la puntuación máxima que esta parte de la pantalla ha cumplido con sus necesidades y la otra mitad de los usuarios con un punto menos.

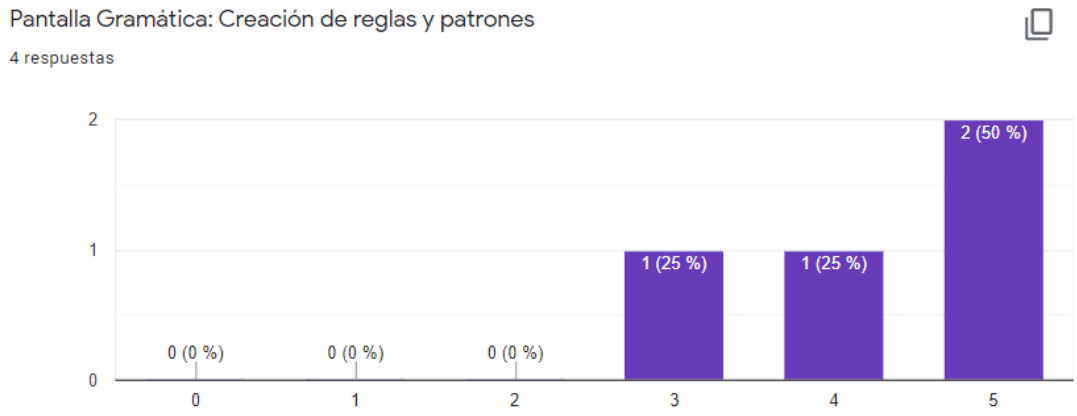


Figura 54: Resultados evaluación con usuarios: Pregunta 3

En la Figura 54, se pregunta por la pantalla de gramática, sobre el editor de reglas y patrones. La mitad de los usuarios han valorado con la puntuación máxima el editor de reglas y patrones. Un usuario con un punto menos y otro usuario con la mitad de la puntuación.

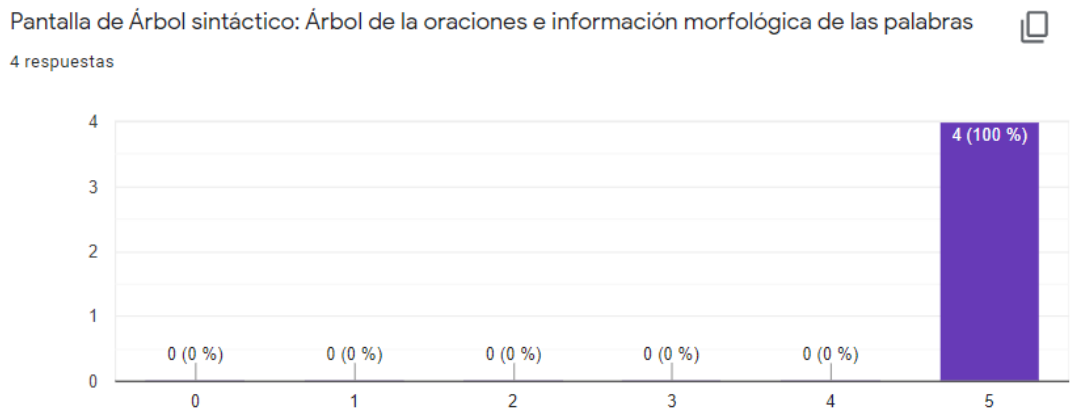


Figura 55: Resultados evaluación con usuarios: Pregunta 4

En la Figura 55 sobre la visualización del árbol sintáctico de las oraciones, todos los usuarios han valorado esta parte de la herramienta con la máxima puntuación.

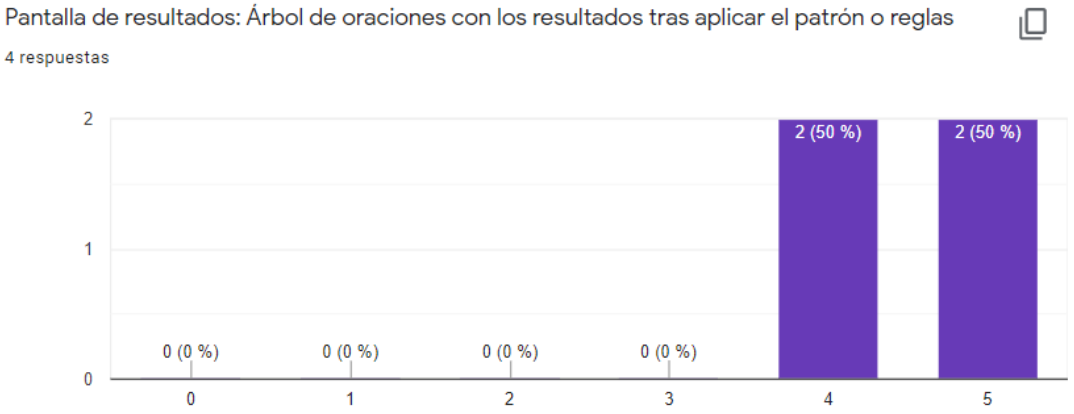


Figura 56: Resultados evaluación con usuarios: Pregunta 5

En la Figura 56, la mitad de los usuarios han valorado con la puntuación máxima la pantalla de resultados. La otra mitad de los usuarios la han valorado con un punto menos.

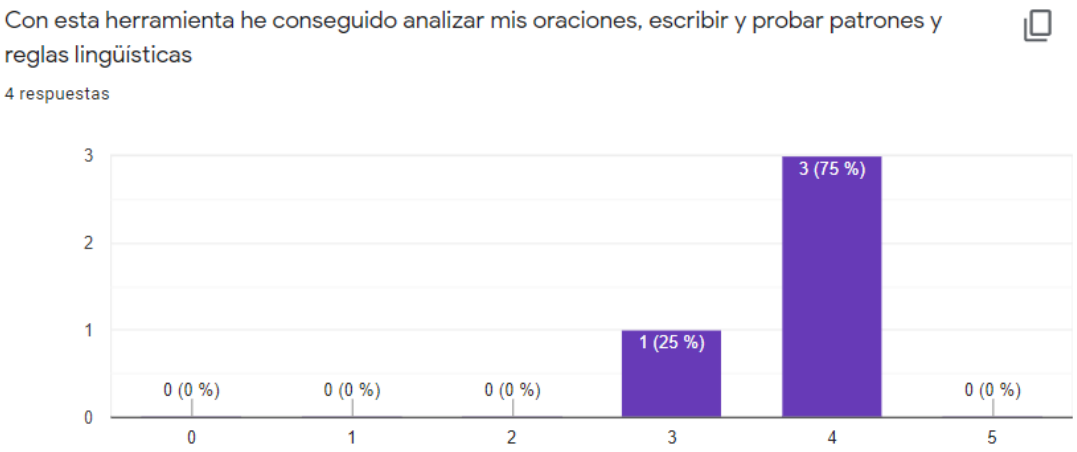


Figura 57: Resultados evaluación con usuarios: Pregunta 6

En la Figura 57, se pregunta por la funcionalidad de aplicar patrones y reglas a los textos interesados. Tres de los usuarios encuestados han valorado positivamente la experiencia con la segunda puntuación más alta y uno de ellos con la mitad de la puntuación.

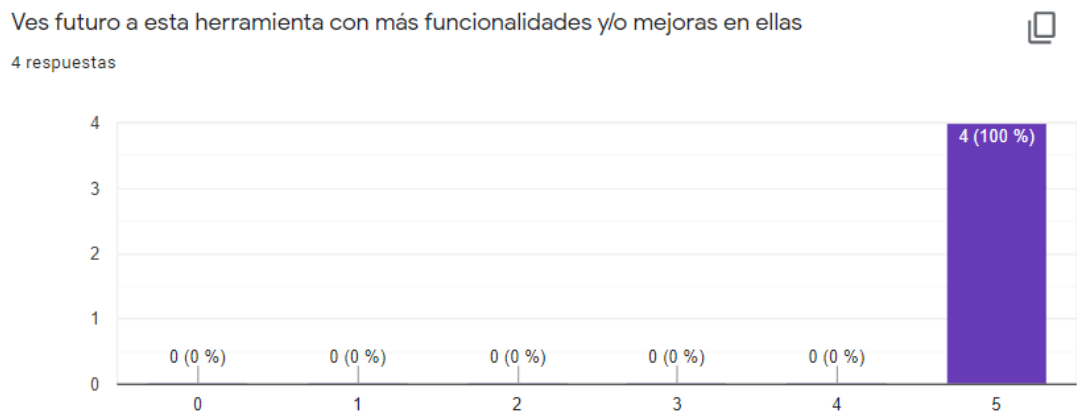


Figura 58: Resultados evaluación con usuarios: Pregunta 7

Finalmente en la Figura 58, se pregunta al usuario por el futuro de la herramienta, y todos los usuarios han valorado positivamente un futuro en el desarrollo de la aplicación con posibles mejoras.

Capítulo 7: Conclusiones y trabajo futuro

7.1. Conclusiones

Con respecto a los objetivos planteados al inicio del proyecto, se han alcanzado y cumplido los objetivos que se propusieron. Fundamentalmente se ha cumplido desarrollar una aplicación web, para que pueda ser utilizada por cualquier tipo de usuario, que necesite analizar un texto y probar patrones y reglas. Para ayudar al usuario con la herramienta, se ha pensado en dos tipos de vistas, una primera pensada para usuarios con mayores conocimientos, podrán realizar cualquier tipo de patrón lingüístico, para que se busque en el cualquier corpus, y/o se puedan escribir reglas para reducir las palabras de una oración del corpus cargado. Y otra vista más sencilla y pensada para los usuarios que no poseen conocimientos de la nomenclatura utilizada para la escritura de patrones y reglas. En ella los usuarios podrán crear reglas para la reducción de oraciones, mediante un formulario. Otra característica que se pensó, fue ayudar a los usuarios a determinar las palabras más importantes en una oración, es por ello que la forma en la que se muestran las oraciones es en un forma de árbol de dependencias de las palabras de la oración.

Para valorar el cumplimiento de los objetivos, el proyecto fue probado por algunos usuarios voluntarios para que mostrasen su opinión de la herramienta. Dando una conclusión positiva de la herramienta y de las funciones que se pueden realizar. Argumentando además posibles mejoras para el futuro de la aplicación. Se concluye por tanto, que este proyecto tiene la capacidad de análisis que otros proyectos profesionales como Grew o los descritos en la sección *1.3*, siendo además una primera aproximación sobre como se puede explotar la automatización del análisis de los textos. La forma que la herramienta consigue realizar las operaciones sobre la búsqueda de patrones y aplicación de reglas en los corpus lingüísticos, es haciendo uso y recogiendo los resultados de la herramienta Grew, indispensable debido a la infinidad de posibilidades de patrones y posibles regla que un usuario puede llegar a crear. Pero el lado negativo de esta herramienta fue la dificultad de instalación de Grew ya que no en cualquier sistema operativo es valido su uso, y a pesar de su instalación no todas las funciones que presenta funcionan correctamente.

7.2. Trabajo futuro

Una posible ampliación relacionada con el desarrollo de las reglas, sería aumentar la vista simple y realizar un formulario con mayores posibilidades de creación de reglas, y así ayudar a los usuarios que no conozcan la nomenclatura utilizada en el editor de texto, para la aplicación de reglas. Aunque este trabajo conllevaría analizar todas las posibilidades en la creación de las mismas, por lo que habría que valorar su viabilidad.

Uno de los aspectos más comentados por los usuarios, en el formulario que realizaron tras la prueba de la aplicación, fue que se pudieran crear reglas utilizando otra nomenclatura que no fuera Ocaml, que es la que se utilizada en Grew. Se podría modificar y crear un propio lenguaje de reglas, y así presentar al usuario varias formas de creación de reglas. Además los resultados se muestran como un grafo arbóreo. Estos resultados son estáticos y no pueden modificarse, a menos que se aplique otra regla. Un trabajo futuro en este punto sería poder modificar estos árboles pulsando sobre ellos para eliminar palabras, acortarlos o conectar palabras con otras.

Otras posibles modificaciones futuras de la herramienta, podrían ser añadir nuevas funciones, tales como realizar el análisis sintáctico de las oraciones y mostrarse visualmente al usuario, o incluso determinar el lema y rema de cada oración.

Chapter 8: Introduction

In this first chapter, the motivation of the project, the objectives set and the work plan followed to organize the research and implementation of the project will be presented.

8.1. Motivation

An essential task of philology has been and continues to be the study of words, their origin, their morphology, to which grammatical category they belong, if they are adjectives, nouns, if they are verbs, what prepositions exist ... the knowledge about all the types of words that exist, the next step was to put different words together and form sentences with coherence and cohesion, and proceed to the study of how words depend on each other. For example, finding the subject, the predicate, the direct, indirect complement ... Traditionally this task has been called how to carry out a syntactic analysis of sentences. The study of observing how words depend on each other in sentences is a necessary process to find the key meaning of sentences and focus on the fundamental message that a text wants to convey.

This process can be very simple, in case you have to analyze only one sentence of a few words, or very long and expensive if you have to analyze several texts that make up many words. This difficulty is due to the fact that for each sentence you have to analyze each of the words and study how they depend on each other. Once this analysis process has finished, you can perform any functionality you want, such as simplifying the text by eliminating unnecessary propositions, searching how many times different words are repeated or finding out the theme of the text, you can even get to know what thoughts have been get to have the author to write the text.

This project focuses on the analysis of any text, in order to help in this expensive process. Automating tasks such as analysis and dependencies of the words that make up the text. Thanks to this, the user will be able to invest the time directly in doing everything they had planned to do after having analyzed it.

8.2. Objectives

The main objective in this project has been to develop a web application to analyze any linguistic corpus that contains a set of texts, and offer the user the possibility of searching for patterns, making linguistic rules and graphically representing the results. This main objective can be specified in several sub-objectives:

- Design an intuitive and simple tool aimed at both users with linguistic knowledge, and users who lack this knowledge, in two different types of interfaces.
- Allow any user to load any type of corpus, in different languages, and in different formats.
- Show the user the results found during the analysis, and the types of sentences loaded.
- Store the morphological information of all the words of the loaded texts.
- Enable a section for the user with linguistic knowledge, in order to write patterns and thus carry out a search in the analyzed corpus, and inform the user of the results obtained after this process.
- Grant the user an editor so that he can write linguistic rules, apply them to the sentences of the loaded corpus and thus be able to reduce the words that make up a sentence.
- Visualize the sentences as a graph in the form of a dependency tree, where the words of the sentence will be the nodes, and observe all the morphological information of each word.

- Save and undo the operations performed on each sentence.
- Download the entire corpus analysis process carried out at any time.
- Offer a simple form to perform the creation of some rules to simplify sentences.

8.3. Workplan

For the development of the project, the following work plan has been carried out:

1. Research on similar tools. In this first phase, a search was made for applications that carried out work similar to that of the project, to take as reference different aspects of design and functionality.
2. Requirements specification. In this phase, all the functionalities that were going to be implemented in the project were determined.
3. Design of the application. In this phase the sketches of how it was thought, the elements and the user interface would be designed.
4. Development of implementation and design. In this phase the tool was developed with all the proposed functionalities and an intuitive and visual interface.
5. Tests and final evaluation. During this phase functional tests were carried out to detect errors and problems, and to be able to solve them.

Figure 1 shows the schedule of the tasks to be carried out for the development of the project and an estimate of the dates.

Chapter 9: Conclusions and future work

9.1. Conclusions

Regarding the objectives set at the beginning of the project, the objectives set have been reached and met. Fundamentally, it has been fulfilled to develop a web application, so that it can be used by any type of user, who needs to analyze a text and test patterns and rules. To help the user with the tool, two types of views have been thought, a first thought for users with more knowledge, they can make any type of linguistic pattern, so that any body is searched for, and/or rules can be written to reduce the words of a sentence of the loaded corpus. And another view that is simpler and intended for users who do not have knowledge of the nomenclature used for writing patterns and rules. In it, users can create rules for reducing sentences using a form. Another feature that was thought of was helping users determine the most important words in a sentence, which is why the way sentences are displayed is in a tree form of dependencies on the words of the sentence.

To assess the fulfillment of the objectives, the project was tested by some voluntary users to show their opinion of the tool. Giving a positive conclusion of the tool and the functions that can be performed. Also arguing possible improvements for the future of the application. Therefore, it is concluded that this project has the capacity of analysis than other professional projects such as Grew or those described in section *1.3*, being also a first approximation on how the analysis analysis of texts can be exploited. The way that the tool manages to perform the operations on the search for patterns and application of rules in the linguistic corpus, is by using and collecting the results of the Grew tool, indispensable due to the infinity of possibilities of patterns and possible rules that a user can create. But the negative side of this tool was the difficulty of installing Grew since its use is not valid in any operating system, and despite its installation, not all the functions it presents work correctly.

9.2. Future work

A possible extension related to the development of the rules, would be to increase the simple view and make a form with greater possibilities of creating rules, and thus help users who do not know the nomenclature used in the text editor, for the application of rules. Although this work would involve analyzing all the possibilities in creating them, so their viability should be assessed.

One of the aspects most commented on by users, in the form they made after testing the application, was that rules could be created using another nomenclature that was not Ocaml, which is the one used in Grew. You could modify and create your own rule language, and thus present the user with various ways of creating rules. Furthermore the results are shown as a tree graph. These results are static and cannot be modified, unless another rule applies. A future job at this point would be to be able to modify these trees by clicking on them to remove words, shorten them or connect words with others.

Other possible future modifications of the tool could be adding new functions, such as performing the syntactic analysis of the sentences and showing them visually to the user, or even determining the motto and row of each sentence.

Bibliografía

[1] Definición de análisis sintático - http://132.248.9.195/ptd2009/marzo/0641731/0641731_A1.pdf - Accedido el día 12/3/20

[2] Definición de corpus ligüistico - https://iris.unito.it/retrieve/handle/2318/57837/6964 - Accedido el día 12/3/20

[3] Características de los corpus ligüisticos - http://diposit.ub.edu/dspace/handle/2445/66774 - Accedido el día 12/3/20

[4] Artículo sobre el análisis sintáctico de las oraciones - https://dialnet.unirioja.es/servlet/articulo?codigo=5279857 - Accedido el día 15/3/20

[5] Trabajo sobre Creación de un treebank de dependencias universales - http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/5334 - Accedido el día 16/3/20

[6] Definición de la gramatica de dependencias - https://es.qwe.wiki/wiki/Dependency_grammar - Accedido el día 20/3/20

[7] Artículo sobre un analizador sintáctico estadístico basado en dependencias para el euskera - http://ixa.si.ehu.es/sites/default/files/dokumentuak/3224/ARTsepln-Kepa-Koldo.pdf - Accedido el día 15/3/20

[8] Codificación (anotación y etiquetado) de los córpora - http://elies.rediris.es/elies18/233.html - Accedido el día 24/3/20

[9] Tesis doctoral de aplicaciones de procesamiento del lenguaje natural y recuperación de información - https://www.researchgate.net/publication/307174605_Representing_Layered_and_Structured_Data_in_the_CoNLL-ST_Format - Accedido el día 12/3/20

[10] Artículo sobre la anotación de dependencia universal para análisis multilenguaje - https://www.aclweb.org/anthology/P13-2017.pdf - Accedido el día 22/3/20

[11] Información sobre CoNLL - https://www.aclweb.org/anthology/W06-2920.pdf - Accedido el día 22/3/20

[12] Página sobre el formato oficial para el etiquetado ConLL-U - https://universaldependencies.org/format.html - Accedido el día 22/3/20

[13] Artículo sobre el uso de Freeling - https://upcommons.upc.edu/handle/2117/15986 - Accedido el día 25/3/20

[14] Dependecias universales CONLL-U - http://arademaker.github.io/files/propor-2018-demo2.pdf - Accedido el día 22/3/20

[15] Herramienta Grew - http://grew.fr/ - Accedido el día 20/3/20

[16] Herramienta online Grew-match - http://match.grew.fr/ - Accedido el día 20/3/20

[17] Información y uso del lenguaje OCaml - https://sunsite.icm.edu.pl/packages/ocaml/ocaml-4.08/ocaml-4.08-refman.pdf - Accedido el día 20/3/20

[18] Herramienta online Grew-parse - http://parse.grew.fr/ - Accedido el día 20/3/20

[19] Información sobre uso Grew-match y etiquetado - https://hal.inria.fr/hal-02267475/ - Accedido el día 20/3/20

[20] Instalación y uso de la herramienta Tred - http://ufal.mff.cuni.cz/tred/ - Accedido el día 1/4/20

[21] Página web DgAnnotator información de herramienta y descarga - http://medialab.di.unipi.it/Project/QA/Parser/DgAnnotator/ - Accedido el día 1/4/20

[22] Página web información de uso Treex - http://ufal.mff.cuni.cz/treex - Accedido el día 1/4/20

[23] Herramienta Treex online - https://lindat.mff.cuni.cz/services/treex-web/run - Accedido el día 1/4/20

[24] Página web de información, uso, descarga y prueba online Brat - http://brat.nlplab.org/ - Accedido el día 1/4/20

[25] UDpipe información, uso y prueba online - http://lindat.mff.cuni.cz/services/udpipe/ - Accedido el día 1/4/20

[26] Página web desarrolladora de proyectos PNL - https://turkunlp.org/projects.html - Accedido el día 1/4/20

[27] Transformación de patrones arbóreos Tregex, Tsurgeon and Semgrex - https://www.aclweb.org/anthology/W17-6528.pdf - Accedido el día 3/4/20

[28] Herramienta online Deep search http://bionlp-www.utu.fi/dep_search/ - Accedido el día 1/4/20

[29] Uso de patrones mediante Tsurgeon - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.380.3530&rep=rep1&type=pdf - Accedido el día 5/4/20

[30] Uso de patrones mediante Semgrex - https://nlp.stanford.edu/nlp/javadoc/javanlp-3.5.0/edu/stanford/nlp/semgraph/semgrex/SemgrexPattern.html - Accedido el día 5/4/20

[31] Herramienta Pyconll para la edición del formato CoNLL - https://pyconll.readthedocs.io/en/stable/ - Accedido el día 6/4/20

[32] Artículo sobre la metodología ágil Scrum - http://sutlib2.sut.ac.th/sut_contents/H129174.pdf - Accedido el día 5/3/20

[33] Arquitectura cliente-servidor - https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf - Accedido el día 5/3/20

[34] Información sobre HTML5 - https://books.google.es/books?hl=es&lr=&id=szDMlRzwzuUC&oi=fnd&pg=PA1&dq=html5&ots=0BuGR_wFQb&sig=g0wF4VsJMmtQLhKPpO_hUTgP040&redir_esc=y#v=onepage&q=html5&f=false - Accedido el día 10/4/20

[35] Definición DOM - http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=pdm&paperid=576&option_lang=eng - Accedido el día 10/4/20

[36] Información sobre CSS3 - https://books.google.es/books?hl=es&lr=&id=2P-4ff445ZcC&oi=fnd&pg=PA20&dq=css3&ots=o44biqIgJG&sig=9g4RLApRmw2DDOEnUVowxx9gmzE&redir_esc=y#v=onepage&q=css3&f=false - Accedido el día 10/4/20

[37] Artículo sobre PHP - https://books.google.es/books?hl=es&lr=&id=zMK3GOMOpQ4C&oi=fnd&pg=PR17&dq=php&ots=FgosW1Jiun&sig=HL3-MFV75wDDIHw3WKWN-_O-sY8&redir_esc=y#v=onepage&q=php&f=false - Accedido el día 10/4/20

[38] Herramienta Javascript - http://190.57.147.202:90/xmlui/bitstream/handle/123456789/430 - Accedido el día 10/4/20

[39] Uso Jquery - https://books.google.es/books?hl=es&lr=&id=gDhVZ35PA8AC&oi=fnd&pg=PT16&dq=jquery&ots=ops6_3J6qZ&sig=wUcJggbh2Z3h-5UazcmIOJUKsKI&redir_esc=y#v=onepage&q=jquery&f=false - Accedido el día 10/4/20

[40] Página web python - https://ieeexplore.ieee.org/abstract/document/4160250 - Accedido el día 12/4/20

[41] Página oficial PhpMyAdmin - https://www.phpmyadmin.net/ - Accedido el día 11/4/20

[42] Utilización MySQL - http://justpain.com/eBooks/Databases/MySQL/ - Accedido el día 11/4/20

[43] Información lenguaje SQL - https://www.bortzmeyer.org/sql-standard.pdf - Accedido el día 15/4/20

[44] Artículo sobre HTTP Apache - https://ieeexplore.ieee.org/abstract/document/612229 - Accedido el día 15/4/20

[45] Comparación de JSON con XML- https://www.cs.montana.edu/izurieta/pubs/caine2009.pdf - Accedido el día 15/4/20

[46] Página oficial de uso Sublime Text 3 - https://www.sublimetext.com/3 - Accedido el día 15/4/20

[47] Herramienta para el etiquetado léxico y análisis sintáctico de textos orientado a la construcción de corpus supervisados - https://rua.ua.es/dspace/bitstream/10045/1879/1/PLN_26_18.pdf - Accedido el día 20/3/20

[48] Artículo sobre la creación de un treebank de dependencias universales mediante recursos existentes para lenguas próximas: el caso del gallego - http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/5334/3118 - Accedido el día 13/5/20

[49] Construcción de un corpus etiquetado sintácticamente para el euskera - https://rua.ua.es/dspace/bitstream/10045/1662/1/PLN_29_01.pdf - Accedido el día 19/5/20

[50] Extensiones para instalar en la herramienta Tred - http://ufal.mff.cuni.cz/tred/extensions/ - Accedido el día 16/3/20

Apéndices

11.2. Apéndice A: Manual de instalación

Actualmente el proyecto está instalado en un servidor perteneciente a la Facultad de Ingeniería Informática para que cualquier usuario pueda probar sus funcionalidades sin necesidad de la instalación de la herramienta. Se puede acceder al proyecto a partir del siguiente enlace http://147.96.240.110/TFM/

11.2.1. Instalación local

Para la instalación local de la herramienta, será necesario acceder a todo el contenido del proyecto a partir de la carpeta habilitada https://drive.google.com/drive/folders/1hdVUAwzX16bBIJFYKvWZkbt5HQly11Y_?usp=sharing . En dicha carpeta se pueden encontrar todos los archivos necesarios para su uso, y algunos archivos para la prueba de la herramienta, como corpus lingüísticos, reglas o patrones. Además se presenta un archivo .sql con la base de datos y los datos de la aplicación web.

A continuación se desarrollan los recursos necesarios para la prueba de la herramienta:

11.2.1.1 Página web

Será necesario descargar la herramienta Xampp, para poder acceder tanto a la base de datos, como al servidor HTTP de manera local. La página para descargar dicha herramienta es https://www.apachefriends.org/es/download.html. A continuación habrá que crear una base de datos e importar el contenido del archivo .sql adjunto del proyecto. El enlace para poder acceder a la base de datos es https://www.phpmyadmin.net/. Finalmente habrá que copiar la carpeta raíz del proyecto (TFM) con todos sus archivos en la ruta donde se haya instalado Xampp, en la carpeta */opt/lampp/htdocs/*. Hasta este punto se podrá acceder al proyecto con el siguiente enlace: http://localhost:8080/TFM

11.2.1.2 Python

Para el uso de la herramienta es necesario la instalación de python 3.5 Se podrá descargar esta versión en https://tutorial.djangogirls.org/es/python_installation/.

11.2.1.3 Grew

Finalmente, será necesario la instalación de la herramienta Grew, dicha herramienta puede ser instalada en Mac OS, Windows y Linux. Para el desarrollo del proyecto fue instalada en Linux, debido a que la herramienta en Windows está en fase beta. Para la instalación de dicha herramienta y sus dependencias es necesario seguir la siguiente guía http://grew.fr/install/. También serán utilidad estos dos enlaces en caso de ocurrir un error durante la instalación: http://grew.fr/upgrade/ y https://opam.ocaml.org/doc/Install.html.

Si al hacer uso de la aplicación, se observan errores, es recomendable realizar los siguientes pasos:

- Encontrar la instalación de *grewpy, con which grewpy* y *grew, con which grew.*
- A continuación se creará un enlace simbólico en /usr/bin, con:
  - *cd /usr/bin*
  - *sudo ln -s* (salida del comando *which grew/grewpy*)

11.3. Apéndice B: Manual de usuario

En este manual se describe detalladamente las funcionalidades que puede realizar el usuario en la aplicación web. Para ello además se mostrarán todas las pantallas que se pueden visualizar.

11.3.1. Encabezado de las pantallas

El encabezado es igual en todas las pantallas de la aplicación y tiene dos variantes, si se ha iniciado sesión se visualizará la Figura 59

Figura 59: Pantalla principal de la aplicación - Sesión iniciada

Los componentes enumerados son de la Figura 61 son:

1. Cambiar vista: es un botón que redirige al usuario a la otra vista. Si el usuario está en la vista avanzada (se detalla en la sección *Vista avanzada para realizar patrones y reglas*), le redirigía a la vista simple de creación de reglas (se detalla en la sección *Vista simple para la creación de reglas*) y viceversa.

2. Ayuda: es un botón que redirige a la página de ayuda al usuario. Se detalla en la sección *Página de ayuda* toda la información de la página.

3. Documentos guardados: es un botón que redirige a la página de documentos guardados. Se detalla en la sección *Página de documentos guardados* toda las funcionalidades en dicha pantalla.

4. Cerrar sesión: es un botón para cerrar la sesión del usuario.

En cambio si no se ha iniciado sesión los componentes que se verán en el encabezado aparecen en la Figura 60:

Figura 60: Pantalla principal de la aplicación - Sin iniciar sesión

1. Cambiar vista: es un botón con la misma funcionalidad descrita anteriormente.
2. Ayuda: es un botón con la misma funcionalidad descrita anteriormente.
3. Iniciar sesión: es un botón que redirige a la página para iniciar sesión en la aplicación. Se detalla en la sección *Iniciar sesión* toda la información.
4. Registrarse: es un botón que redirige a la página de registrar un usuario. Se detalla en la sección *Registrar usuario* toda la información.

11.3.2. Iniciar sesión

Es una página para que el usuario pueda iniciar sesión en la aplicación, con el usuario y contraseña que el mismo haya creado.

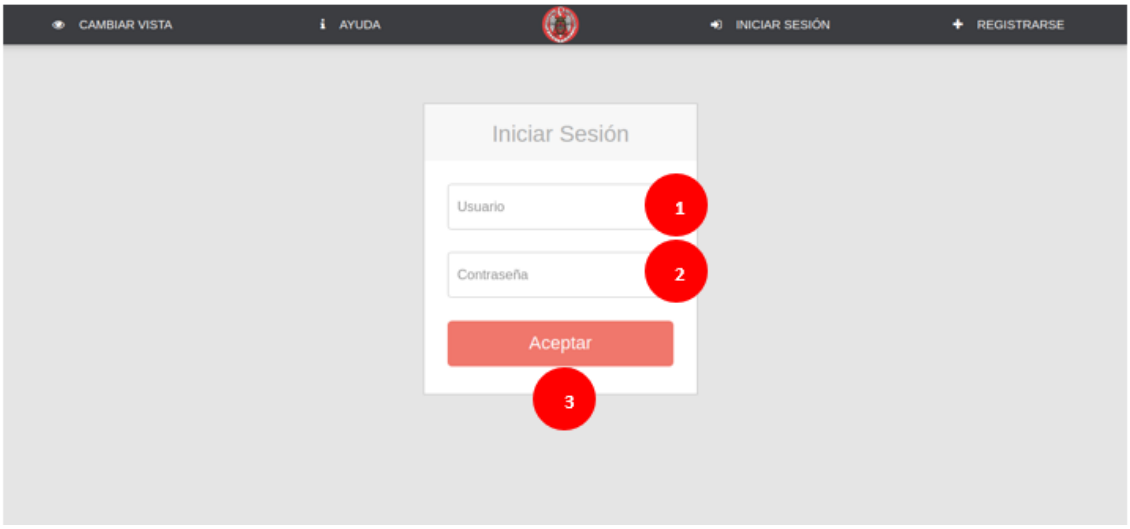


Figura 61: Iniciar sesión en la aplicación

Los elementos que se pueden ver en la Figura 61 son:

1. Es un cuadro de texto donde se espera que el usuario escriba el nombre del usuario dado de alta.
2. Es un cuadro de texto en el que el usuario deberá escribir la contraseña para iniciar su sesión en el sistema.
3. Es un botón para iniciar la sesión, al presionarlo se valida la información introducida. En caso de que no exista el usuario o la contraseña no corresponda a la de ese usuario, se notificará mostrando un mensaje de error.

11.3.3. Registrar usuario

Es una página para que el usuario pueda registrar un usuario en el sistema.

Figura 62: Registrar un usuario en el sistema

Los elementos que se pueden ver en la Figura 34 son:

1. Es un cuadro de texto donde se espera que el usuario escriba el nombre del usuario que quiere dar de alta.
2. Es un cuadro de texto en el que el usuario deberá escribir la contraseña para el usuario que quiere crear.
3. Es un cuadro de texto donde el usuario deberá escribir de nuevo la contraseña. Con el motivo de asegurar al usuario de que no se ha equivocado al introducir la contraseña.
4. Es un botón para registrar el usuario. Al presionarlo se valida la información introducida. En caso de que exista el usuario o que las contraseñas no sean las mismas, se notificará mostrando un mensaje de error.

11.3.4. Página de ayuda

Es una página para responder a ciertas dudas que puedan surgir para la utilización de la herramienta.

Figura 63: Página de ayuda de la aplicación

Los elementos que se pueden ver en la Figura 63 son:

1. Es una sección de un grupo de preguntas sobre un tema concreto.
2. Título de la pregunta, con desplegable para que el usuario pueda pulsar y desplegarse la respuesta.
3. Ejemplo de visualización de pregunta respondida tras pulsar el botón. Además en algunas preguntas hay un enlace para la descarga de corpus, patrones y reglas y facilitar la prueba de la herramienta.

11.3.5. Página de documentos guardados

Es una página con los documentos que hayan guardado los usuarios. Esta pantalla únicamente es accesible si ha iniciado sesión el usuario en la aplicación.

Figura 64: Documentos guardados por un usuario

Los elementos que se pueden ver en la Figura 64 son:

1. Título del documento que haya guardado el usuario se la sesión.
2. Fecha en la que se guardó el documento.
3. Botón para descargar el documento en concreto. Al pulsar se descargará el archivo en formato CoNLL.

11.3.6. Vista avanzada para realizar patrones y reglas

En esta página se podrán realizar múltiples funcionalidades, para desarrollar todas ellas, esta sección se dividirá en 4 subsecciones, cada una de ellas explicando cada parte que podemos visualizar en la Figura 65

Figura 65: Pantalla principal de búsqueda de patrones y aplicación de reglas

El componente común de todas las subsecciones que se explican a continuación es:

1. Es un seleccionable de Pantalla Completa, una vez se pulse en cualquiera de los 4 apartados, esa pantalla aumentará de tamaño y se ajustará a la resolución del ordenador, pensado para visualizar mejor cualquiera de los apartados. Si se quiere volver a la pantalla con el tamaño inicial se deberá pulsar en el mismo seleccionable.

11.3.6.1 Pantalla de corpus

Esta pantalla será una de las primeras en utilizarse cuando se quiera analizar un texto. Está pensada para cargar el corpus lingüístico que desee el usuario y realizar diferentes operaciones que serán descritas a continuación. Al comienzo, esta pantalla, tendrá el aspecto de la Figura 66:

1. Un selector de archivos, el usuario deberá seleccionar un archivo con formato *.conll*
2. Botón para cargar el archivo adjuntado. Se validará el archivo tenga un formato CoNLL-U.

Figura 66: Pantalla de corpus inicialmente

En caso de que el archivo no respete el formato CoNLL-U, aparecerá el mensaje informativo de la Figura 67, para indicar al usuario de que no se ha podido cargar el archivo automáticamente, y debe configurar la información del archivo.

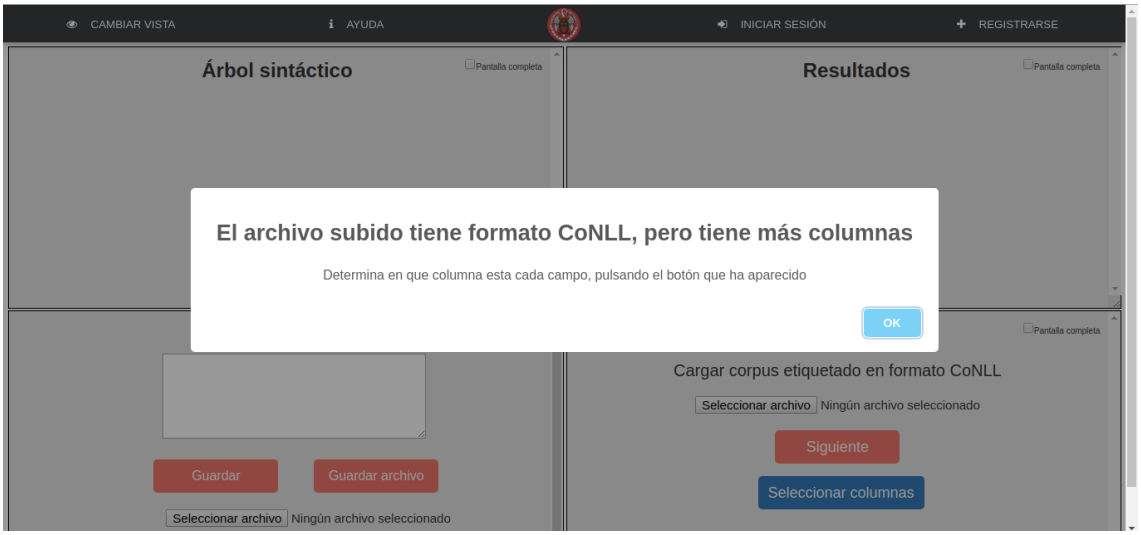


Figura 67: Mensaje informativo de que no se respeta el formato CoNLL-U

Tras mostrar dicho mensaje, el usuario visualizará un formulario para configurar en que columna aparece cada campo del formato ConLL-U (id, lema, forma, UPOS, deprel, núcleo y nombre de la dependencia) en el fichero cargado. Además se muestran las primeras 40 líneas del archivo cargado, para que el usuario sepa la disposición de la información en el archivo. Toda esta información se puede observar en al Figura 68.

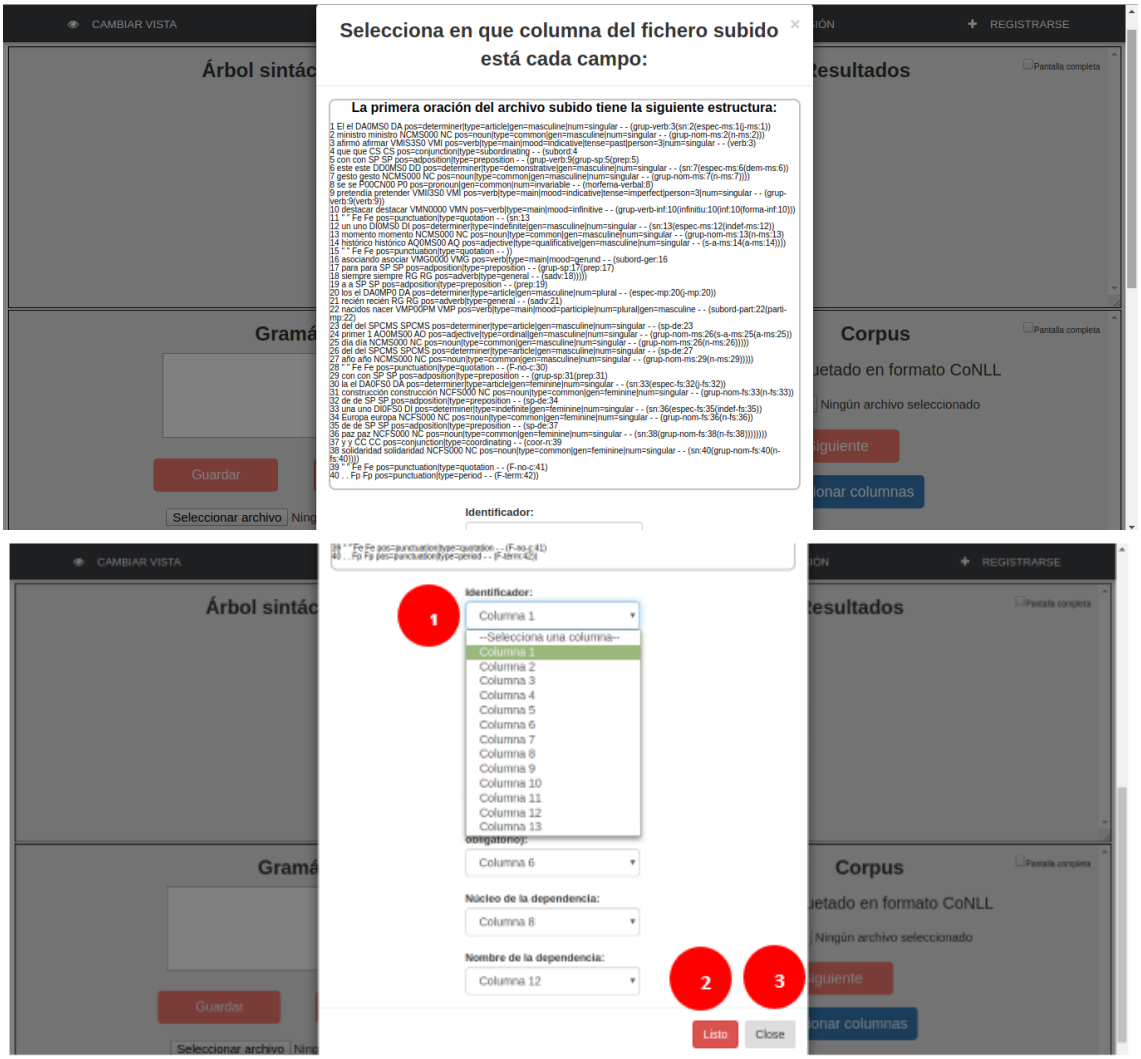


Figura 68: Configurar columnas del archivo cargado

A continuación se describen los componentes de la pantalla:

1. Es un seleccionable con los campos a configurar. El usuario deberá seleccionar en el desplegable, a que columna pertenece cada campo.

2. Es un botón para confirmar la selección de las columnas. Se verificará que se han rellenado todos los campos, a excepción del campo de Rasgos morfológicos", ya que es no obligatorio.

3. Es un botón para cancelar la configuración de los campos.

Una vez el sistema tenga la seguridad de cargar un archivo con formato CoNLL-U, se mostrará al usuario un mensaje informativo como en la Figura 69

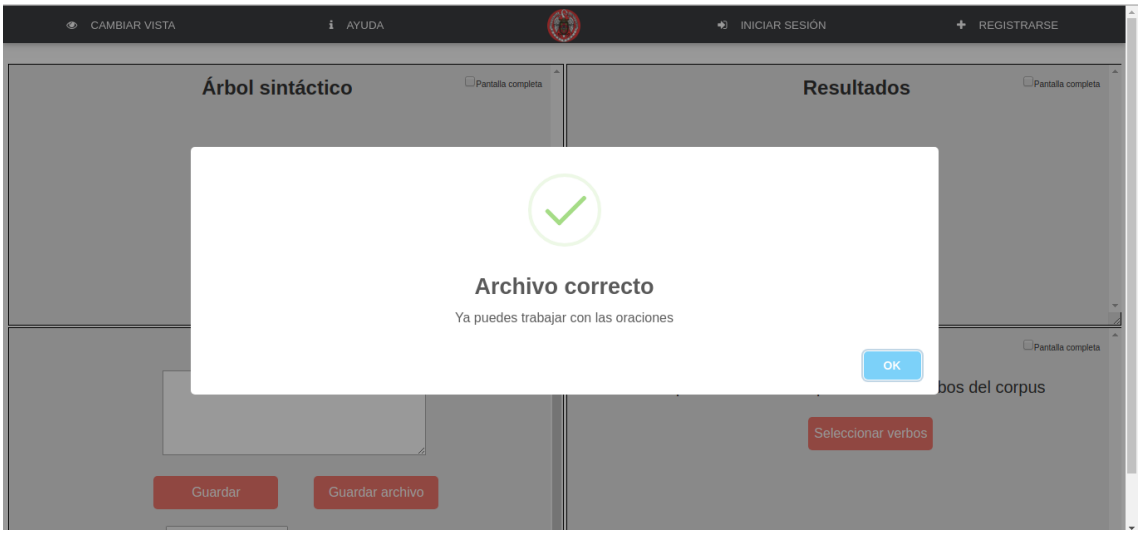


Figura 69: Archivo CoNLL cargado correctamente

Al cerrar el mensaje, se visualizará en la pantalla de corpus como la Figura 70.

Figura 70: Seleccionar las etiquetas verbales en el corpus cargado

El componente que aparece en la nueva pantalla es:

1. Es un botón para que el usuario determine que etiquetas verbales que tienen los verbos del corpus cargado. Al pulsar en el botón se redirigirá a la Figura 71.

Figura 71: Escritura de las etiquetas verbales

En la pantalla de la Figura 71, se mostrará al usuario las 15 primeras oraciones cargadas en el corpus, para tener la visión de que etiquetas gramaticales tiene cada palabra del texto. Deberá escribir en el cuadro de texto los tipos de etiquetas verbales del corpus, con el fin de poder contabilizar cuantos verbos tiene cada oración. Se pueden ver los siguientes componentes en esta página:

1. Es un campo de texto, donde el usuario tendrá que escribir las etiquetas verbales del corpus, cada una de ellas separadas con una coma, y sin importar mayúsculas o minúsculas. Se expone un ejemplo del texto esperado en la marca de agua del texto.

2. Es un botón para finalizar el proceso de escritura de las etiquetas verbales. Una vez pulsado, se contabilizarán cantas etiquetas gramaticales que haya escrito el usuario aparecen por cada oración.

3. Es un botón para cerrar esta ventana emergente y no guardar el proceso.

Una vez realizado este proceso se mostrarán todas las oraciones cargadas, y la información de la Figura 72.

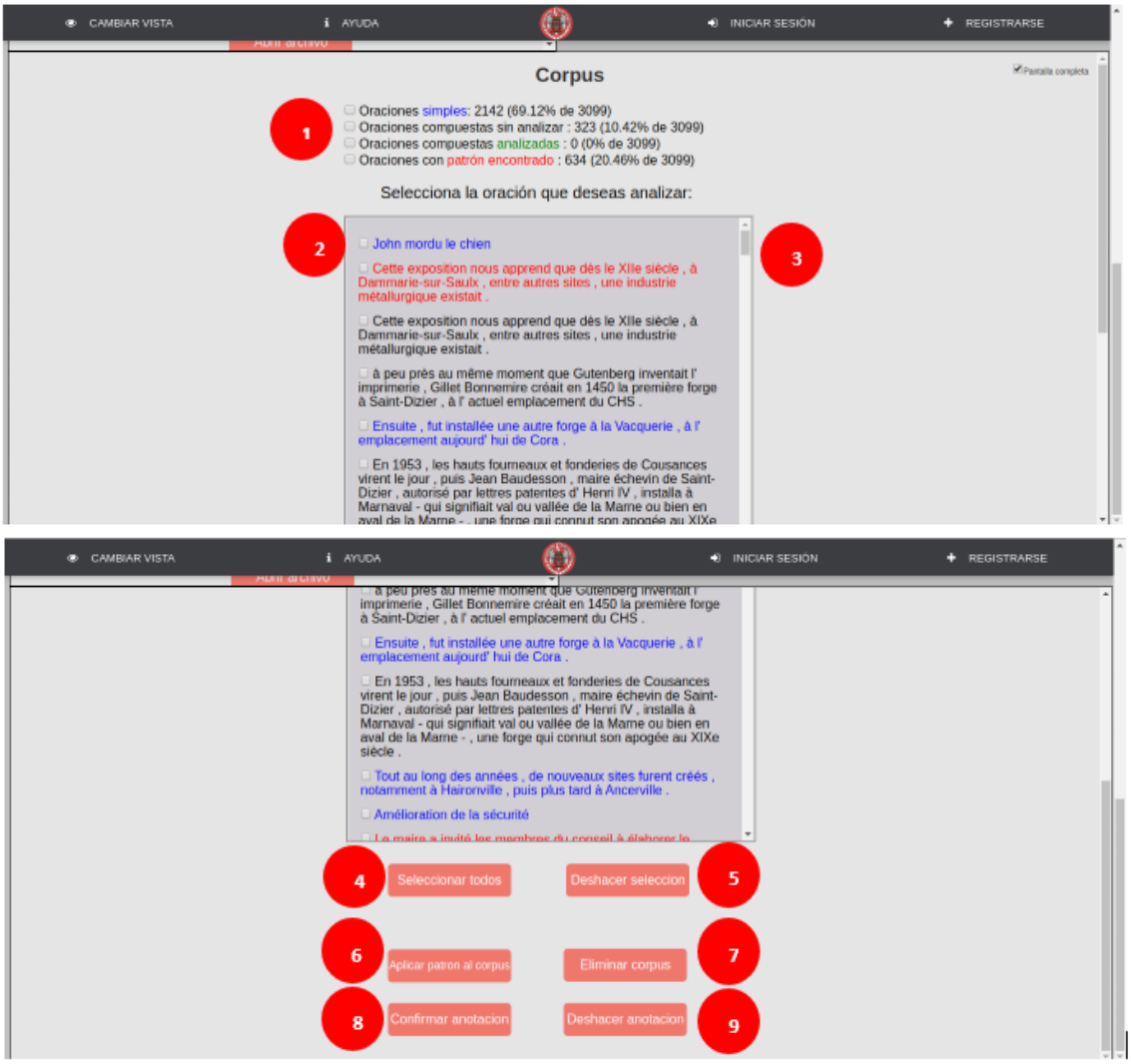


Figura 72: Mostrar oraciones al usuario y funcionalidades en la pantalla corpus

Los componentes que aparecen en estas pantallas son:

1. Información sobre el número de oraciones simples (oraciones con 0 o 1 verbo), compuestas (más de 1 verbo), analizadas (las que tras aplicar una regla a una oración sigue siendo compuesta es decir continua siendo una oración con más de 1 verbo) y oraciones que se ha encontrado el patrón propuesto. Además se habilita un marcaje para cada grupo, y que al pulsarlo se seleccionarán todas las oraciones de este grupo. Por ejemplo si se marca la selección de oraciones simples, se seleccionarán todas las oraciones simples cargadas, el mismo proceso ocurre para las oraciones compuestas, para las analizadas y para las de patrón encontrado.

2. Cada oración del corpus cargado, se habilitará la posibilidad de marcar esa oración en concreto, para realizar las operaciones que se desarrollan a continuación. Además se indica mediante colores el tipo de oración que es al usuario. Azul en caso de que sea simple, negra en caso de ser una oración compuesta, verde si se ha analizado con una regla pero sigue siendo compuesta y roja si se ha encontrado un patrón en esa oración.

3. Al pulsar en cada oración se habilita una pestaña emergente con la información morfológica de las palabras que componen la oración, como se puede observar en la Figura 73.

4. Es un botón para seleccionar todas las oraciones disponibles.

5. Es un botón para quitar todas las selecciones de las oraciones.

6. Es un botón para aplicar el patrón escrito a todas las oraciones del corpus cargado. Se detalla más sobre la escritura de patrones en la sección *Pantalla de Gramática*.

7. Es un botón para eliminar el corpus cargado, dando la posibilidad de seleccionar otro corpus al usuario.

8. Es un botón, utilizado una vez se aplica una regla lingüística a una oración y se quiere guardar los resultados. Para ello se escribirá en el corpus una anotación en la última columna con la palabra "Yes", indicando que esa palabra se mantendrá en la oración y no se va a eliminar en la reducción. Como se puede ver en la Figura 74.

9. Es un botón para deshacer la anotación descrita anteriormente en el corpus.

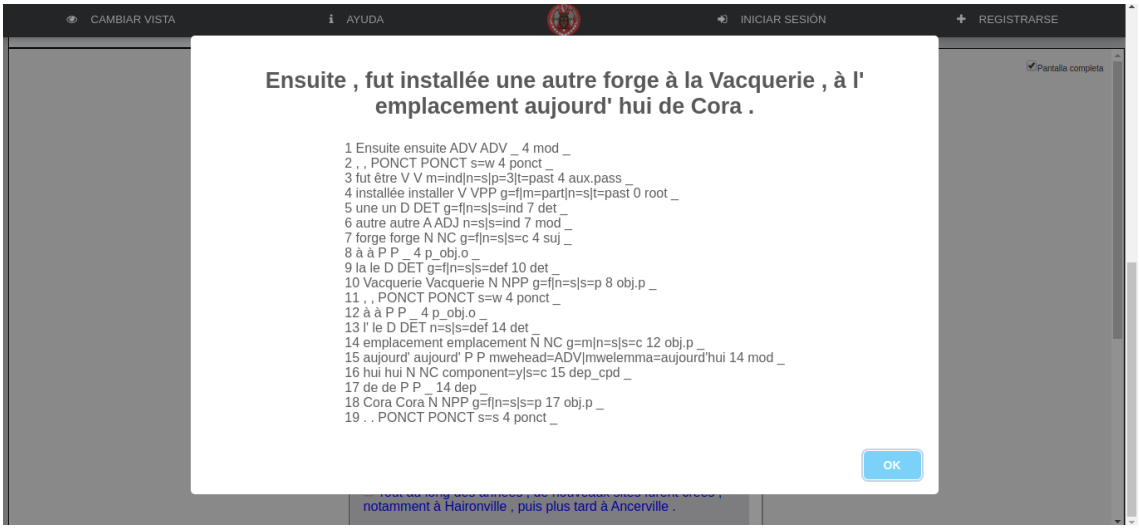
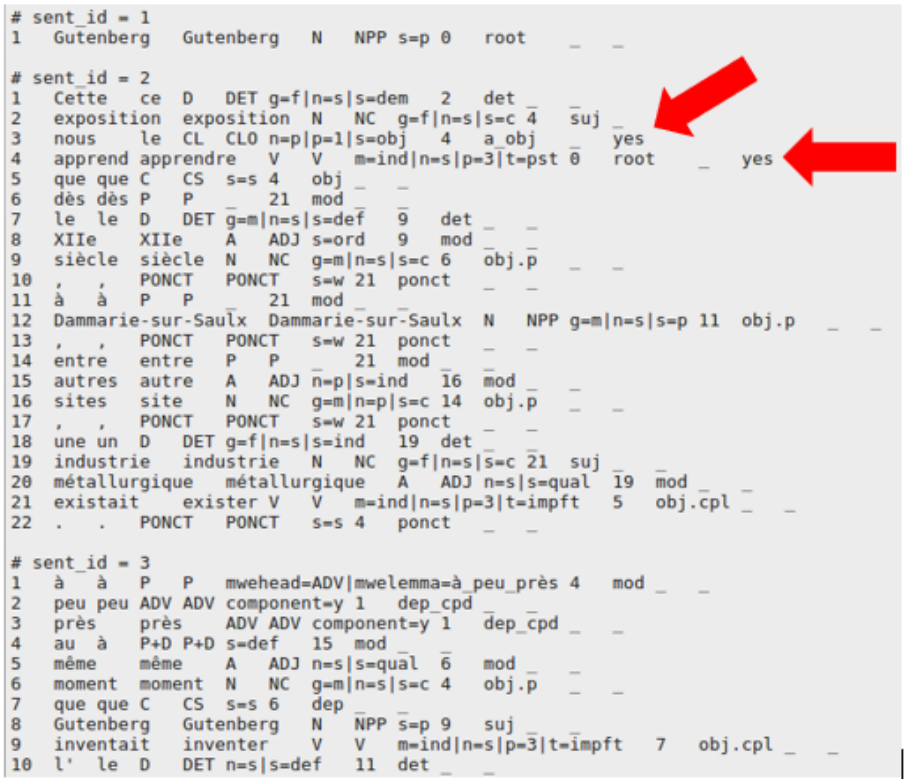


Figura 73: Ventana emergente al seleccionar una oración



```
# sent_id = 1
1   Gutenberg   Gutenberg   N   NPP s=p 0   root    _   _

# sent_id = 2
1   Cette   ce  D   DET g=f|n=s|s=dem   2   det _   _
2   exposition  exposition  N   NC  g=f|n=s|s=c 4   suj _   _
3   nous    le  CL  CLO n=p|p=1|s=obj   4   a_obj   _   yes
4   apprend apprendre   V   V   m=ind|n=s|p=3|t=pst 0   root    _   yes
5   que que C   CS  s=s 4   obj _   _
6   dès dès P   P   _   21  mod _   _
7   le  le  D   DET g=m|n=s|s=def   9   det _   _
8   XIIe    XIIe    A   ADJ s=ord   9   mod _   _
9   siècle  siècle  N   NC  g=m|n=s|s=c 6   obj.p   _   _
10  ,   ,   PONCT   PONCT   s=w 21  ponct   _   _
11  à   à   P   P   _   21  mod _   _
12  Dammarie-sur-Saulx  Dammarie-sur-Saulx  N   NPP g=m|n=s|s=p 11  obj.p   _   _
13  ,   ,   PONCT   PONCT   s=w 21  ponct   _   _
14  entre   entre   P   P   _   21  mod _   _
15  autres  autre   A   ADJ n=p|s=ind   16  mod _   _
16  sites   site    N   NC  g=m|n=p|s=c 14  obj.p   _   _
17  ,   ,   PONCT   PONCT   s=w 21  ponct   _   _
18  une un  D   DET g=f|n=s|s=ind   19  det _   _
19  industrie   industrie   N   NC  g=f|n=s|s=c 21  suj _   _
20  métallurgique   métallurgique   A   ADJ n=s|s=qual  19  mod _   _
21  existait    exister V   V   m=ind|n=s|p=3|t=impft   5   obj.cpl _   _
22  .   .   PONCT   PONCT   s=s 4   ponct   _   _

# sent_id = 3
1   à   à   P   P   mwehead=ADV|mwelemma=à_peu_près 4   mod _   _
2   peu peu ADV ADV component=y 1   dep_cpd _   _
3   près    près    ADV ADV component=y 1   dep_cpd _   _
4   au  à   P+D P+D s=def   15  mod _   _
5   même    même    A   ADJ n=s|s=qual  6   mod _   _
6   moment  moment  N   NC  g=m|n=s|s=c 4   obj.p   _   _
7   que que C   CS  s=s 6   dep _   _
8   Gutenberg   Gutenberg   N   NPP s=p 9   suj _   _
9   inventait   inventer    V   V   m=ind|n=s|p=3|t=impft   7   obj.cpl _   _
10  l'  le  D   DET n=s|s=def   11  det _   _
```

Figura 74: Anotación de los resultados del patrón en el corpus

Habrá dos botones que solo aparecerán por dos eventos, como se puede observar en la Figura 75:

- Es un botón que solo aparecerá si se pulsa una o más oraciones, aparecerá la opción para descargar las oraciones seleccionadas en formato .conll
- Este botón solo aparecerá si se pulsa una sola oración. Una vez pulsado se mostrará en la pestaña de *Pantalla de Árbol sintáctico* la oración y su árbol de dependencias.

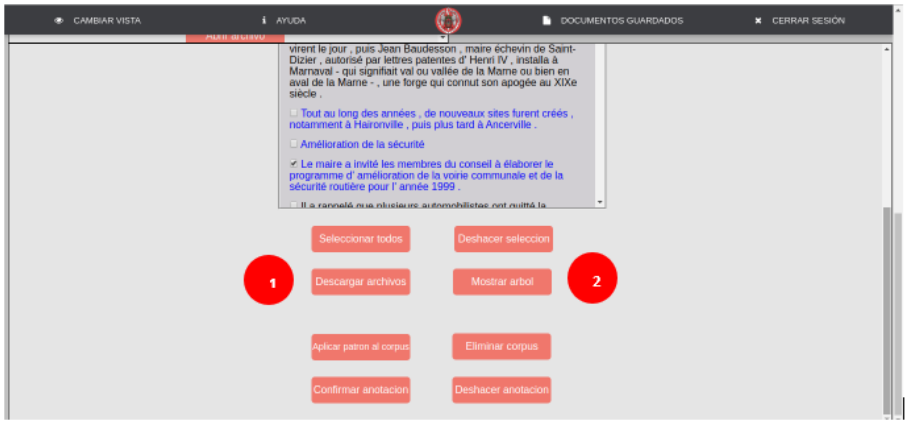


Figura 75: Botones de descargar archivos y mostrar árbol

11.3.6.2 Pantalla de Gramática

Esta pantalla está pensada para que el usuario pueda escribir patrones o reglas, para aplicarlas a las oraciones cargadas en la sección anterior. Los componentes que aparecen en la Figura 76 son:

- Es un editor de texto, donde el usuario podrá escribir las reglas o patrones con la nomenclatura Ocaml.
- Es un botón para que el usuario guarde la regla o patrón que ha escrito en la aplicación.
- Es un botón para guardar la regla o patrón escrito en el editor de una forma local. Al pulsarlo se descargará lo que haya escrito el usuario en el editor.
- Es un selector de archivos, pensado para que el usuario pueda abrir un archivo en el que se encuentre una regla o un patrón que quiera aplicar. Al abrirlo, se copiará el contenido del archivo en el editor para que el usuario pueda verlo y editarlo.

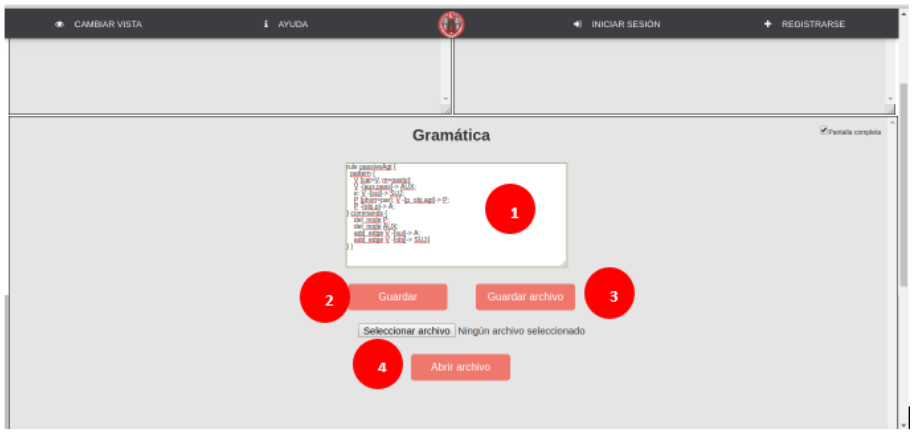


Figura 76: Sección de gramática

11.3.6.3 Pantalla de Árbol sintáctico

En esta pantalla no se visualizará nada hasta que el usuario seleccione una oración del corpus cargado y pulse en el botón de "Mostrar árbol"de la sección *Pantalla de corpus*. Cuando lo haga se mostrará una pantalla como en la Figura 78, donde se mostrará en primer lugar la oración completa, a continuación el número de verbos encontrados en la oración. Y los siguientes componentes:

- Un botón Guardar archivo, solo se visualizará por los usuarios que hayan iniciado sesión. Una vez pulsado el usuario escribirá el nombre con el que quiere guardar el archivo en la base de datos de ese usuario.
- Un enlace para descargar esta oración, junto con la información sintáctica de cada palabra que componen la oración.
- Es un botón para aplicar una regla lingüística a esa oración. Solo se mostrará en caso de que el usuario haya escrito una regla en el editor de texto que aparece en la sección *Pantalla de Gramática*. Si se pulsa dicho botón, y se puede reducir la oración, se mostrarán los resultados en la pestaña de *Visualización de resultados con la aplicación de la regla*. Para indicar al usuario que se ha podido aplicar la regla correctamente se mostrará una pantalla informativa como en la que aparece en la Figura 77

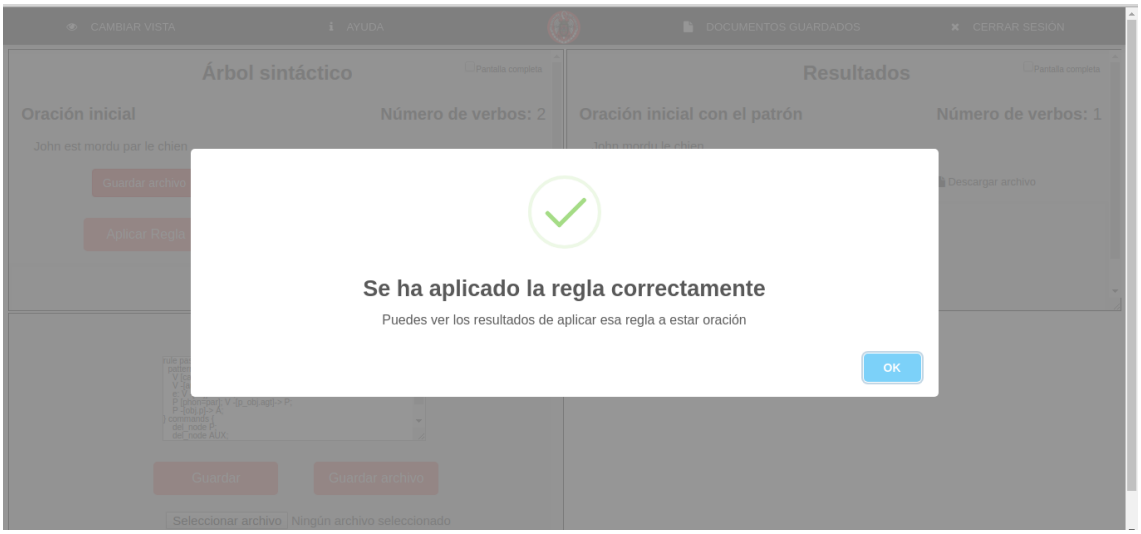


Figura 77: Pantalla informativa de que se ha podido aplicar la regla a la oración seleccionada

- Al pasar el ratón por cualquier nodo que compone el árbol, se mostrará la información morfológica de cada palabra que compone la oración.

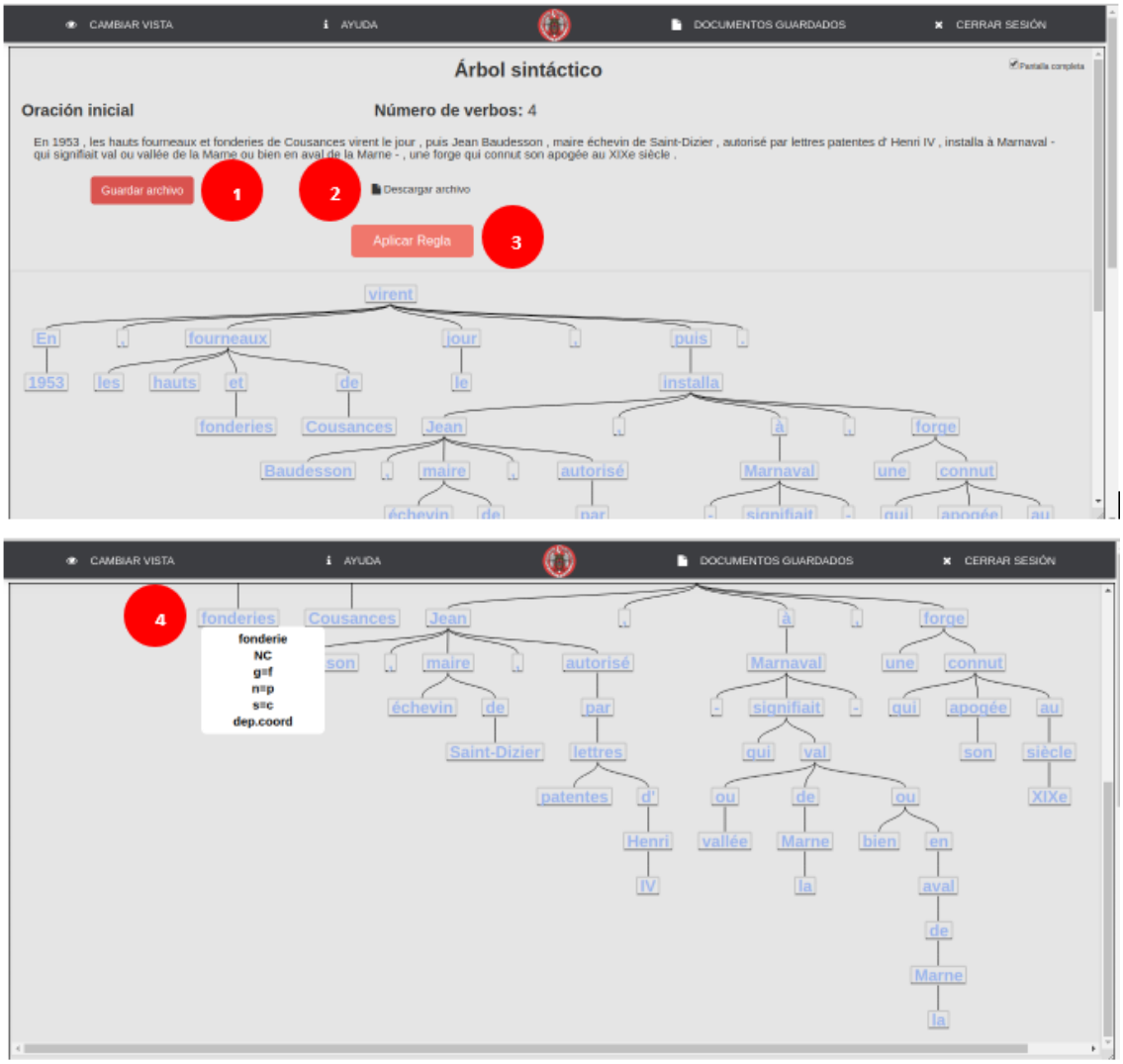


Figura 78: Árbol de dependencias de una oración seleccionada

11.3.6.4 Pantalla de resultados

En esta pestaña se observan los resultados de aplicar una regla a una oración o de el patrón encontrado en una oración. En la Figura 79, se observa que el usuario ha seleccionado una oración, en la pestaña de arriba a la izquierda, y tras pulsar el botón de *"Aplicar regla"*, se reduce la oración a la que se muestra en la pestaña de arriba a la derecha. En este caso se mostrará la oración reducida, el número de verbos de la oración actual tras la reducción y el árbol de dependencias de las palabras. Y que como en la sección anterior también se mostrará la información morfológica de cada palabra al pasar el cursor por cada nodo.

Figura 79: Resultado de aplicar una regla a una oración seleccionada

En caso de aplicar un patrón a la oración, se mostrará una pantalla como en la Figura 80. Se mostrará la oración inicial, pero resaltando las palabras que coincidan en el patrón propuesto. A continuación el número de verbos de la oración y finalmente el árbol de dependencias, exclusivamente de las palabras que coincidan con el patrón. Al igual que anteriormente al pasar el cursor por encima de cualquier nodo se podrá visualizar su información morfológica.

Figura 80: Resultado de comprobar un patrón en una oración

11.3.7. Vista simple para la creación de reglas

En esta página el usuario tendrá un formulario para poder crear reglas lingüísticas, ver las reglas que ha creado y seleccionar un archivo con el corpus lingüístico que quiera analizar, se puede observar en la Figura 81, todos estos elementos.

Figura 81: Página principal de creación de reglas

A continuación se enumeran los componentes de los que se dispone en la página:

1. Es un campo de texto, para que el usuario escriba el nombre de la regla que se va a crear.
2. Es un seleccionable único, donde el usuario indicará que buscará la regla que vaya a crear en el texto cargado.
3. Es un seleccionable único, el usuario determinará que tipo de dependencia corresponde esta regla.
4. Es un marcador para que el usuario seleccione si la palabra buscada es una conjunción o no lo es.
5. Es un marcador donde el usuario selecciona si quiere eliminar la palabra encontrada y las que dependen de ella o no.
6. Es un botón para crear la regla con los campos anteriores. Se validará que el usuario haya iniciado sesión, en caso contrario se mostrará al usuario el error de la Figura 82.
7. Es un selector de archivos, el usuario podrá cargar un archivo local con el corpus que quiera reducir. Solo se podrá subir archivos con formato *.conll*.
8. Es un listado de todas las reglas creadas por el usuario, cada una de ellas será un enlace que redirigía a la página de *Editor de reglas*.
9. Es un botón para validar que el usuario haya cargado un corpus con formato *.conll*, en caso de que ocurra algún error se notificará al usuario, con un mensaje descriptivo en esta misma pantalla. Si el archivo conll presenta un formato CoNLL-U, se accederá a la siguiente pantalla, descrita en la sección *Selector de oraciones del corpus cargado*, si el archivo no tiene formato CoNLL-U, redirigirá a la pantalla descrita en la sección *Reestructuración de los campos del fichero subido*.

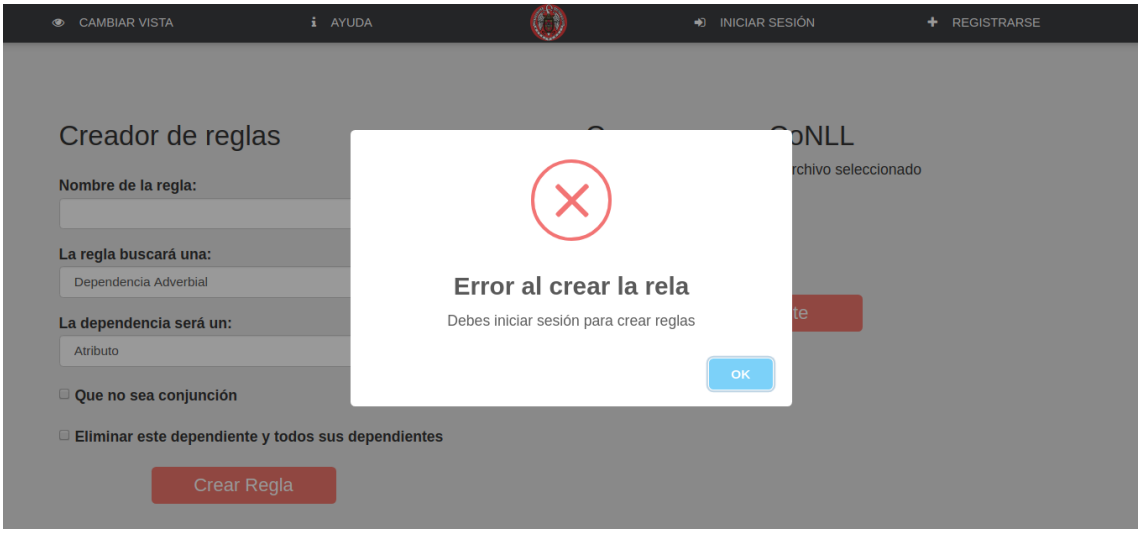


Figura 82: Error al crear una regla, porque no se ha iniciado sesión

11.3.7.1 Editor de reglas

Es una página utilizada para que el usuario pueda editar cualquier campo de la regla que haya creado. Para acceder a dicha página deberá pulsar en una regla de las que aparecen en el listado. En la Figura 83, puede observarse como se presenta el nombre de la regla y las características de la misma, todos los campos menos el nombre son editables.

Figura 83: Editar una regla que haya creado un usuario

Los elementos que contiene la página son:

1. Es un seleccionable único, donde el usuario puede modificar la selección de la palabra a buscar, escogida cuando se creo la regla.

2. Es un seleccionable único, el usuario podrá modificar a que tipo de dependencia corresponde esta regla.
3. Es un marcador para que el usuario puede modificar si la palabra buscada es una conjunción o no lo es.
4. Es un marcador donde el usuario puede modificar si quiere eliminar la palabra encontrada y las que dependen de ella o no.
5. Es un botón para editar esta regla, conllevará modificar los campos alterados por los actuales y la regla se comportará de diferente manera
6. Es un botón para eliminar la regla, al pulsar se eliminará dicha regla del sistema.

11.3.7.2 Reestructuración de los campos del fichero subido

Se indicará al usuario con un mensaje de que el archivo subido no cumple con el estándar, como se observa en la Figura 84:

Figura 84: El corpus cargado tiene más columnas que el formato CoNLL-U

El usuario deberá completar un formulario, para indicar a que columna pertenece cada campo, como puede observarse en la Figura 85 para reestructurar el fichero y que la aplicación pueda tratarlo correctamente.

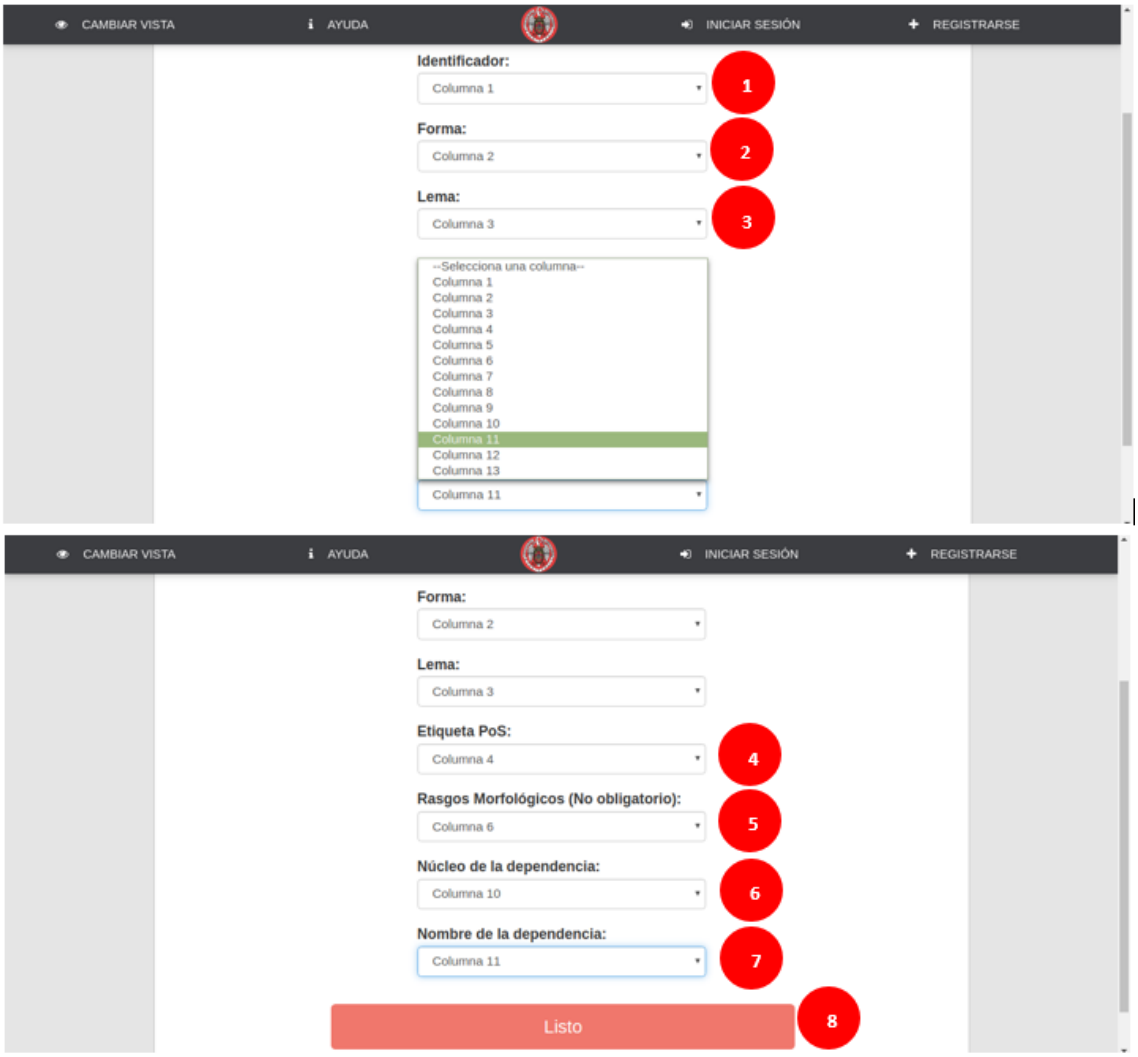


Figura 85: Formulario para indicar a que campo pertenece cada columna del archivo cargado

Los elementos que contiene la pantalla son:

1. Es un desplegable con selector único para indicar en que columna del archivo subido está el campo Id.
2. Es un desplegable con selector único para indicar en que columna del archivo subido está el campo Forma.
3. Es un desplegable con selector único para indicar en que columna del archivo subido está el campo Lema.
4. Es un desplegable con selector único para indicar en que columna del archivo subido está el campo Etiqueta Pos.
5. Es un desplegable con selector único para indicar en que columna del archivo subido está el campo Rasgos Morfológicos, será el único campo que no será necesario de completar.

6. Es un desplegable con selector único para indicar en que columna del archivo subido está el campo núcleo de dependencia.

7. Es un desplegable con selector único para indicar en que columna del archivo subido está el campo nombre de dependencia.

8. Es un botón para confirmar la selección de los campos. Se validará que todos los campos han sido rellenados (menos el de Rasgos Morfológicos), y se modificará el contenido del archivo subido para que presente un formato CoNLL-U, se redirigirá a la sección *Selector de oraciones del corpus cargado*.En caso de que no se hallan rellenado todos los campos se notificará al usuario.

11.3.7.3 Selector de oraciones del corpus cargado

En caso de cargar correctamente el corpus, y se tenga la certeza de que el contenido del fichero tiene formato CoNLL-U, se mostrará al usuario el mensaje correspondiente a la Figura 86.

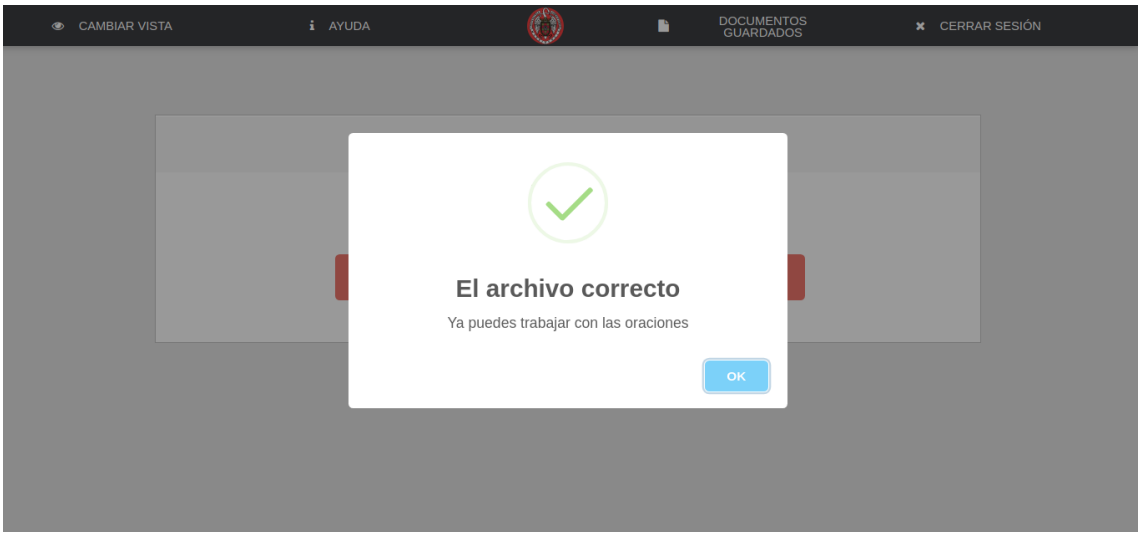


Figura 86: Corpus lingüístico cargado correctamente

A continuación se mostrará al usuario un desplegable con todas las oraciones del corpus para que seleccione una de , como se muestra en la Figura 87, que será la que se analizará con las reglas que posea el usuario.

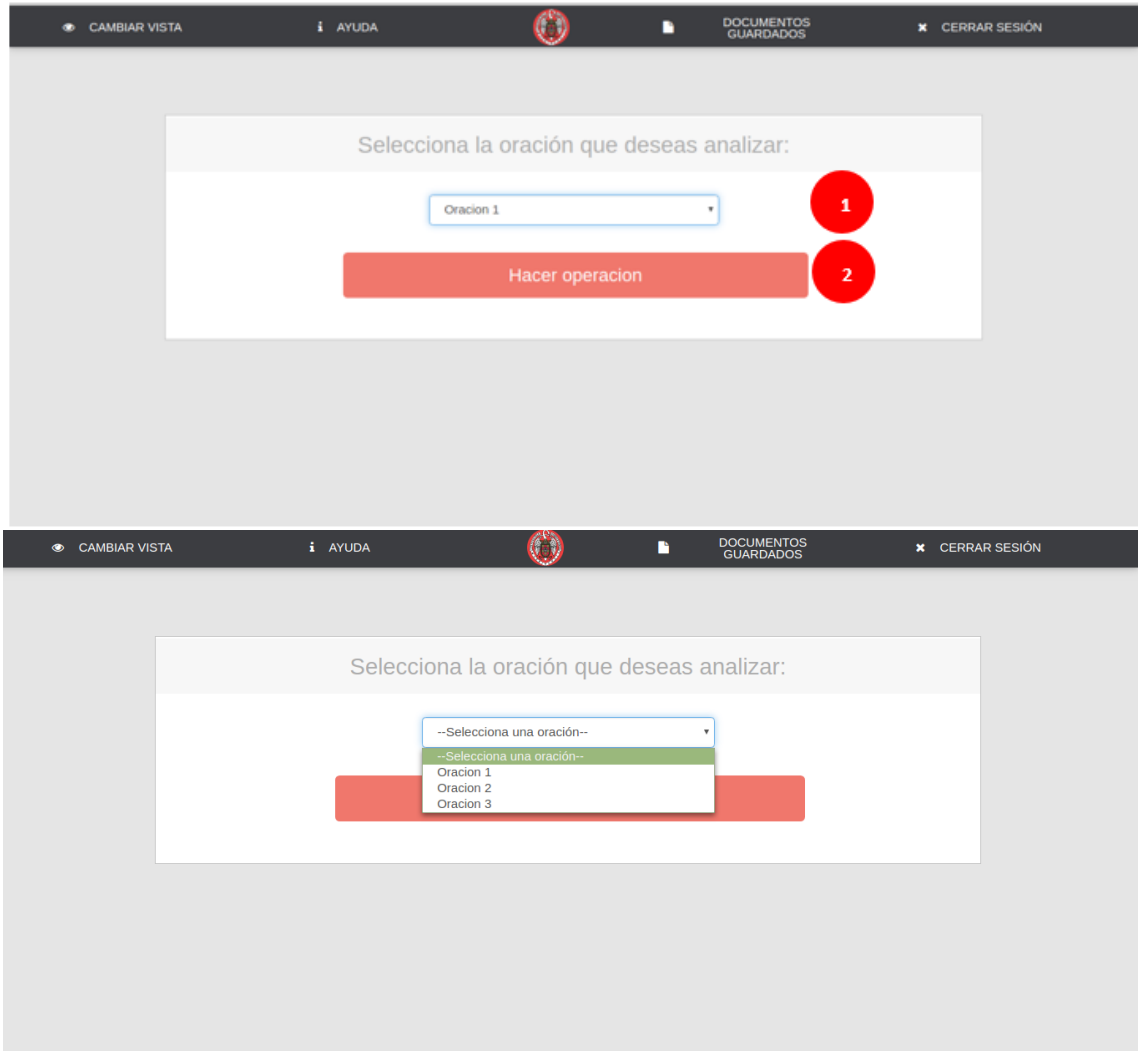


Figura 87: Desplegable con todas las oraciones del corpus lingüístico

Los componentes que se pueden ver son:

1. Es un desplegable de selección única, se seleccionará una oración del listado de oraciones
2. Un botón para confirmar la oración seleccionada. Una vez se pulse, se comprobará que se ha seleccionado una oración y redirigirá a la página descrita en la sección *Visualización de resultados con la aplicación de la regla*.

11.3.7.4 Visualización de resultados con la aplicación de la regla

Una vez seleccionada una de las oraciones del corpus, se mostrará la oración inicial y la oración reducida por cada una de las reglas que haya creado el usuario, en caso de que se haya podido aplicar.

Figura 88: Oración inicial y reducida

En la Figura 88, se pueden observar los siguientes componentes:

1. Botón para guardar la oración en los *Página de documentos guardados* del usuario. Se podrá guardar cualquiera de las dos oraciones, la inicial y/o la reducida y la información morfológica de cada palabra de cada oración con formato CoNLL-U. Solo aparecerá este botón en caso de que el usuario haya iniciado sesión en la aplicación.

2. Es un enlace para descargar la oración inicial y/o reducida con la información morfológica de cada palabra formada en la oración. Se descargará un archivo con formato *.conll*

3. Es un botón para mostrar el árbol de dependencias de la oración inicial o reducida, como se puede observar en la Figura 89. Donde puede observarse la oración representada como un árbol de dependencias, donde cada palabra corresponde a un nodo del árbol. Además se habilita la opción de que al poner el cursor encima de un nodo aparecerá la información morfológica de esa palabra: su etiqueta gramatical, su dependencia, su raíz, género, número, tiempo verbal...

4. Botón para seleccionar otra oración, redirigirá a la pantalla en la sección *Selector de oraciones del corpus cargado*.

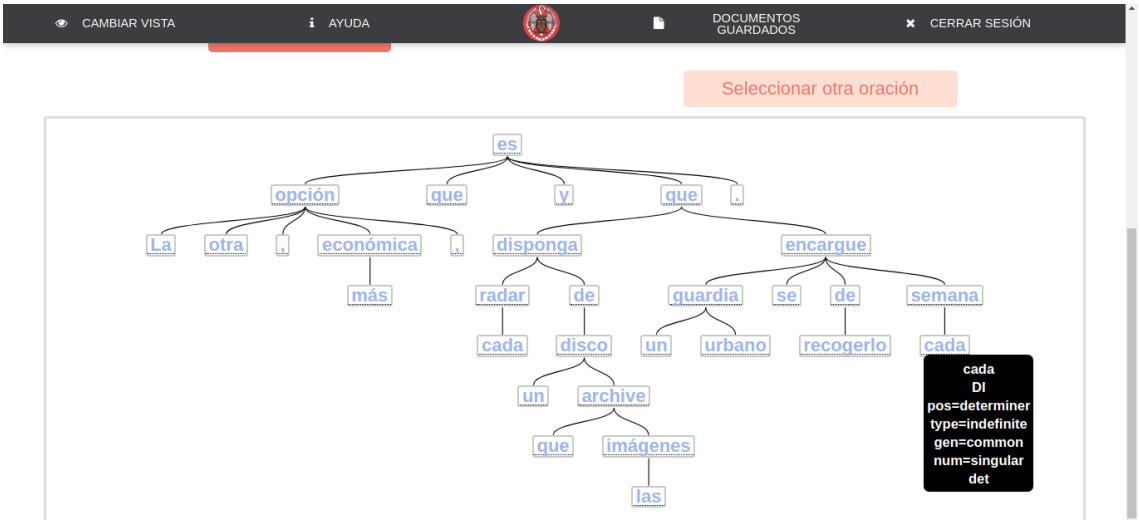


Figura 89: Árbol de dependencias de una oración

Anexo 1: Bocetos de la aplicación

Se adjuntan los bocetos desarrollados manualmente, y comentados por todas las personas vinculadas a este proyecto, con el objetivo fijar los requisitos que presentaría el sistema antes del desarrollo.

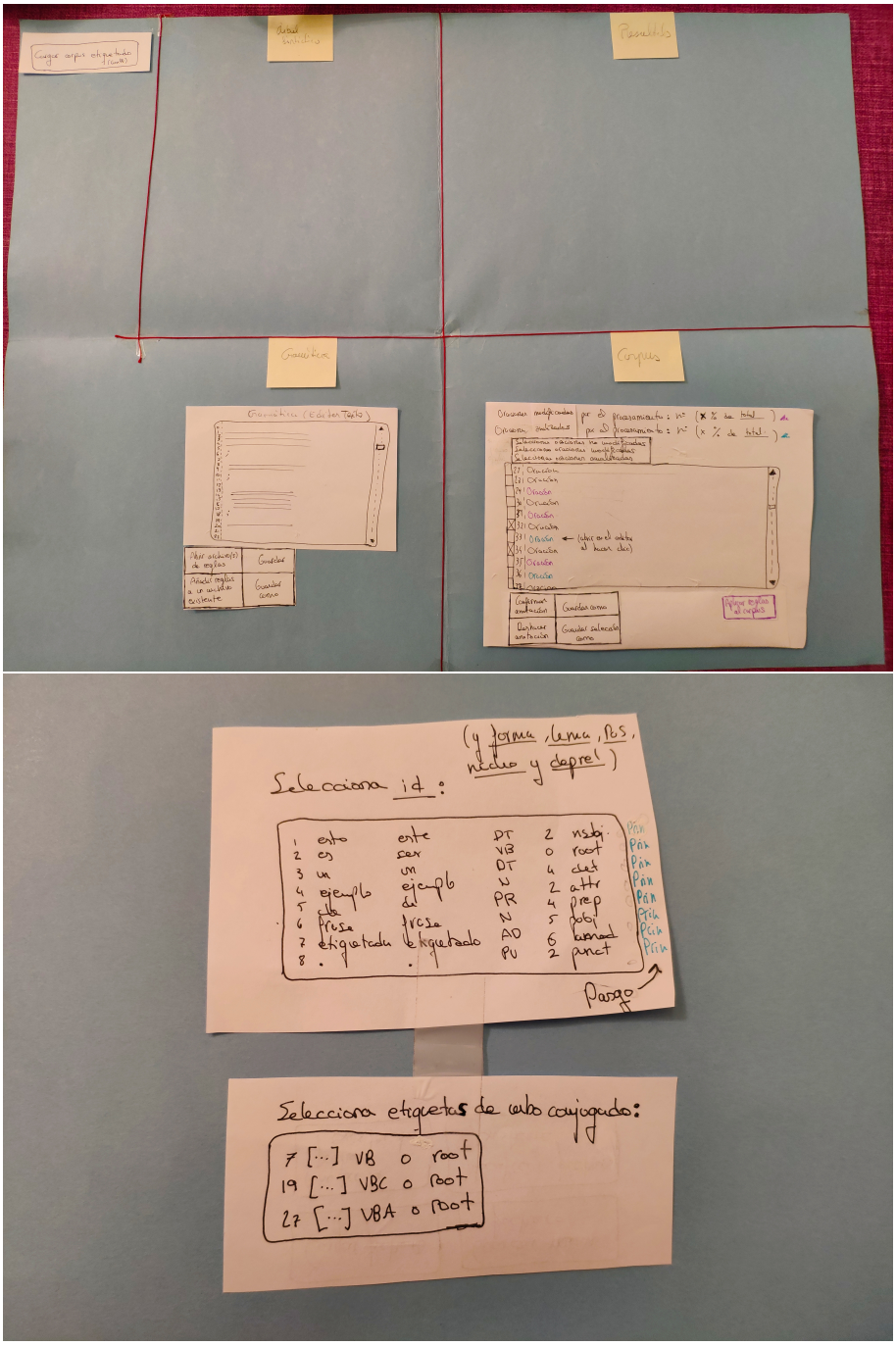


Figura 90: Bocetos de la aplicación

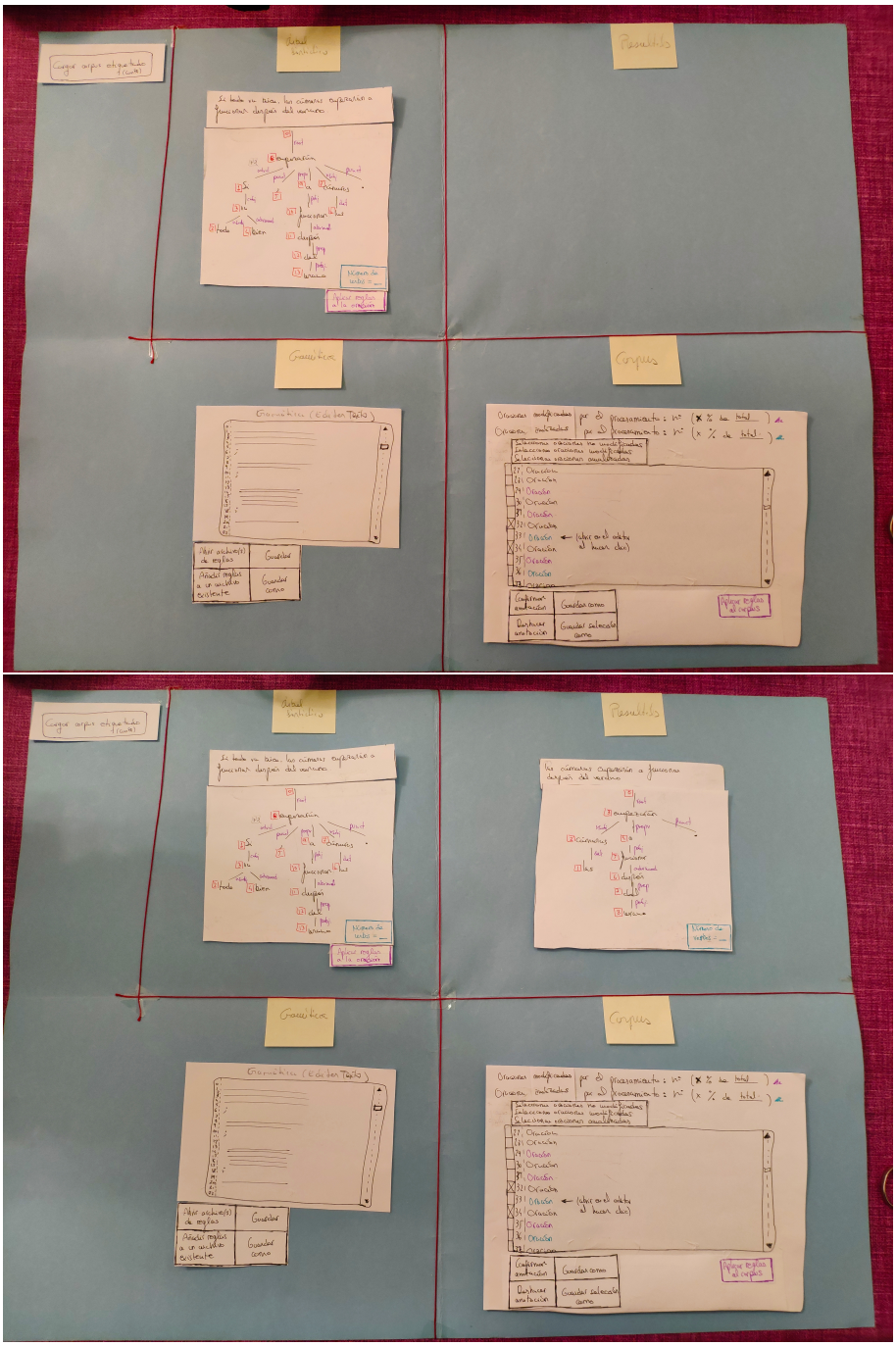


Figura 91: Bocetos de la aplicación

Anexo 2: Preguntas y respuestas de la evaluación con usuarios

A continuación se adjuntan las preguntas realizadas a los usuarios que realizaron la evaluación de la herramienta y sus respuestas en en el primer formulario.

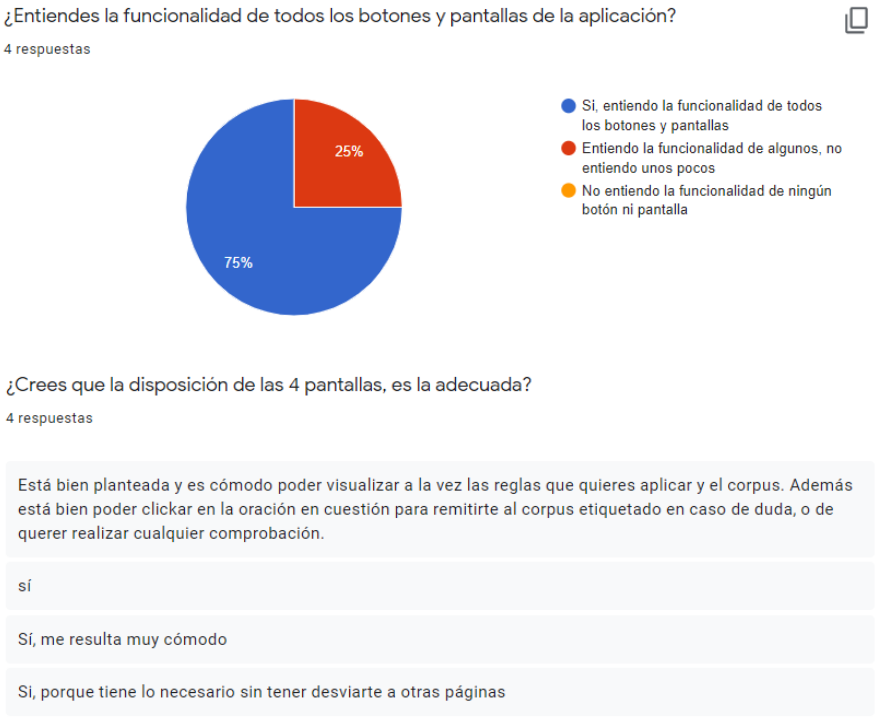


Figura 92: Preguntas y respuestas de la evaluación con usuarios en el formulario 1

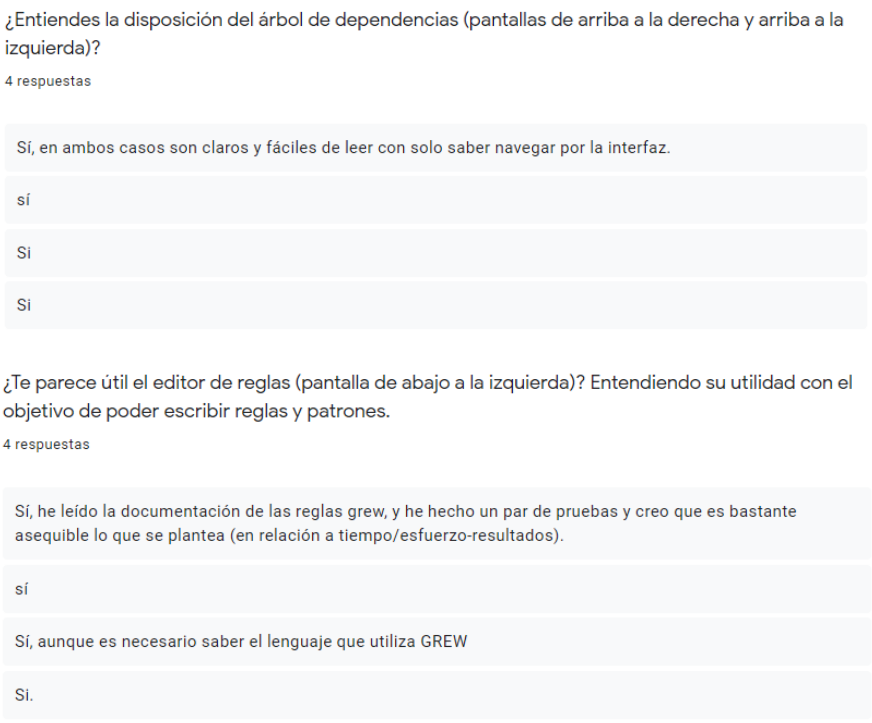


Figura 93: Preguntas y respuestas de la evaluación con usuarios en el formulario 1

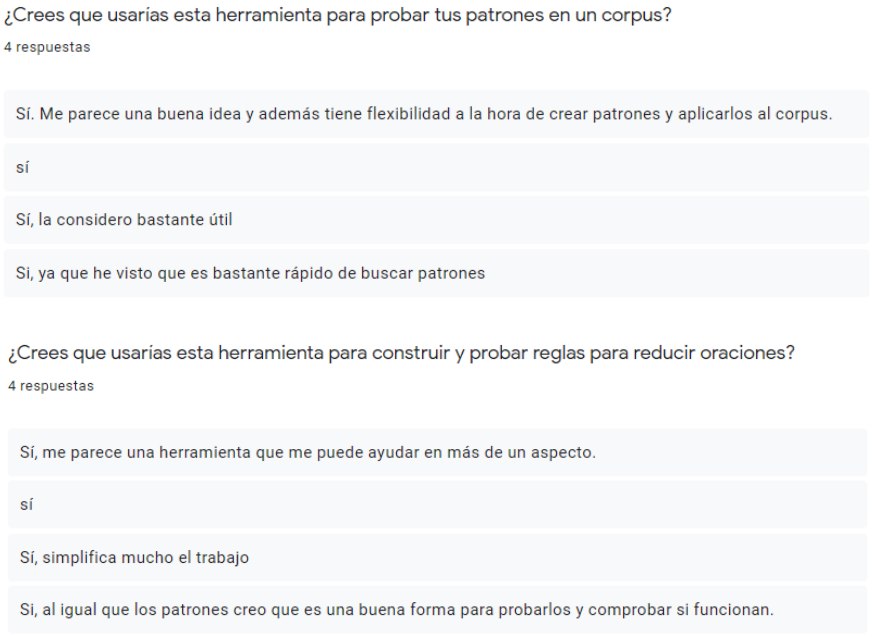


Figura 94: Preguntas y respuestas de la evaluación con usuarios en el formulario 1

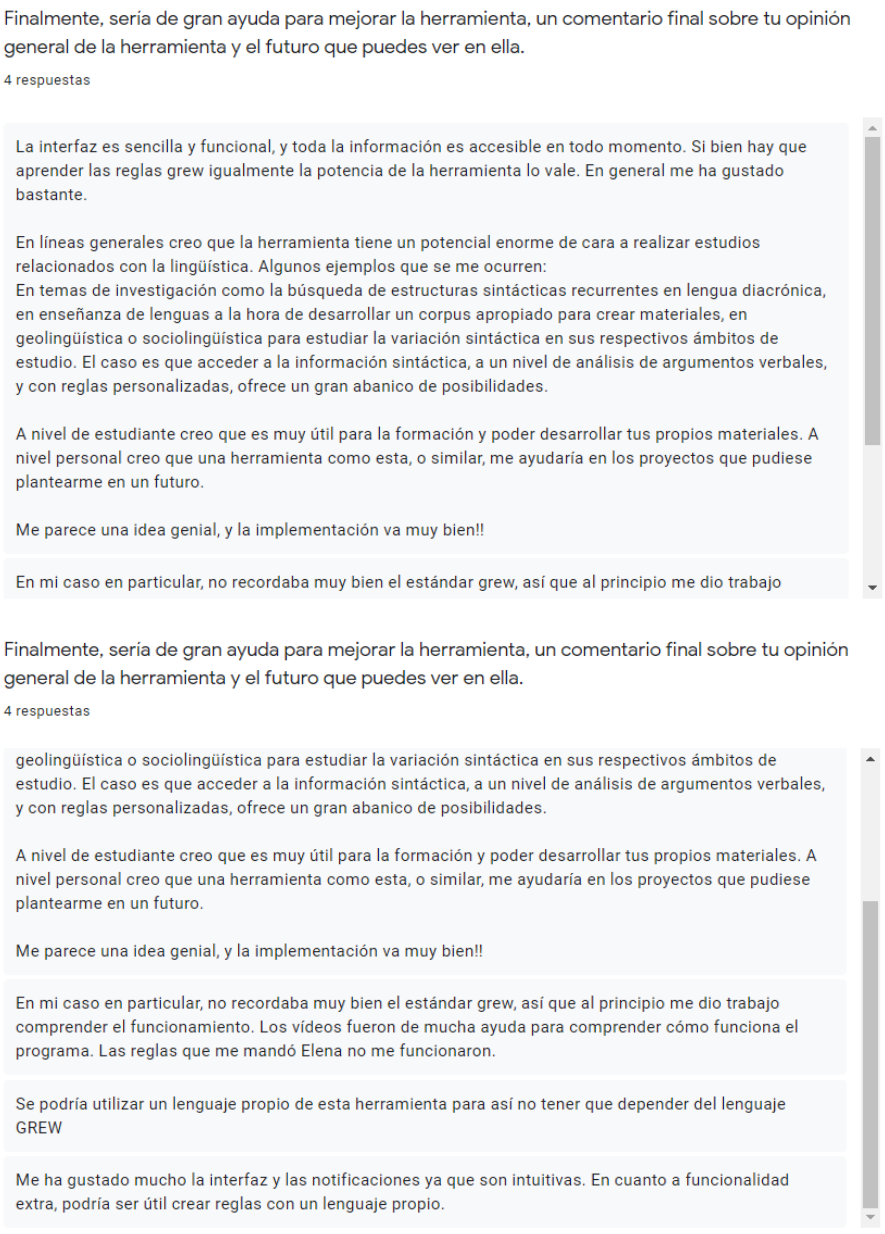


Figura 95: Preguntas y respuestas de la evaluación con usuarios en el formulario 1

También se pueden ver las preguntas propuestas en un segundo formulario.

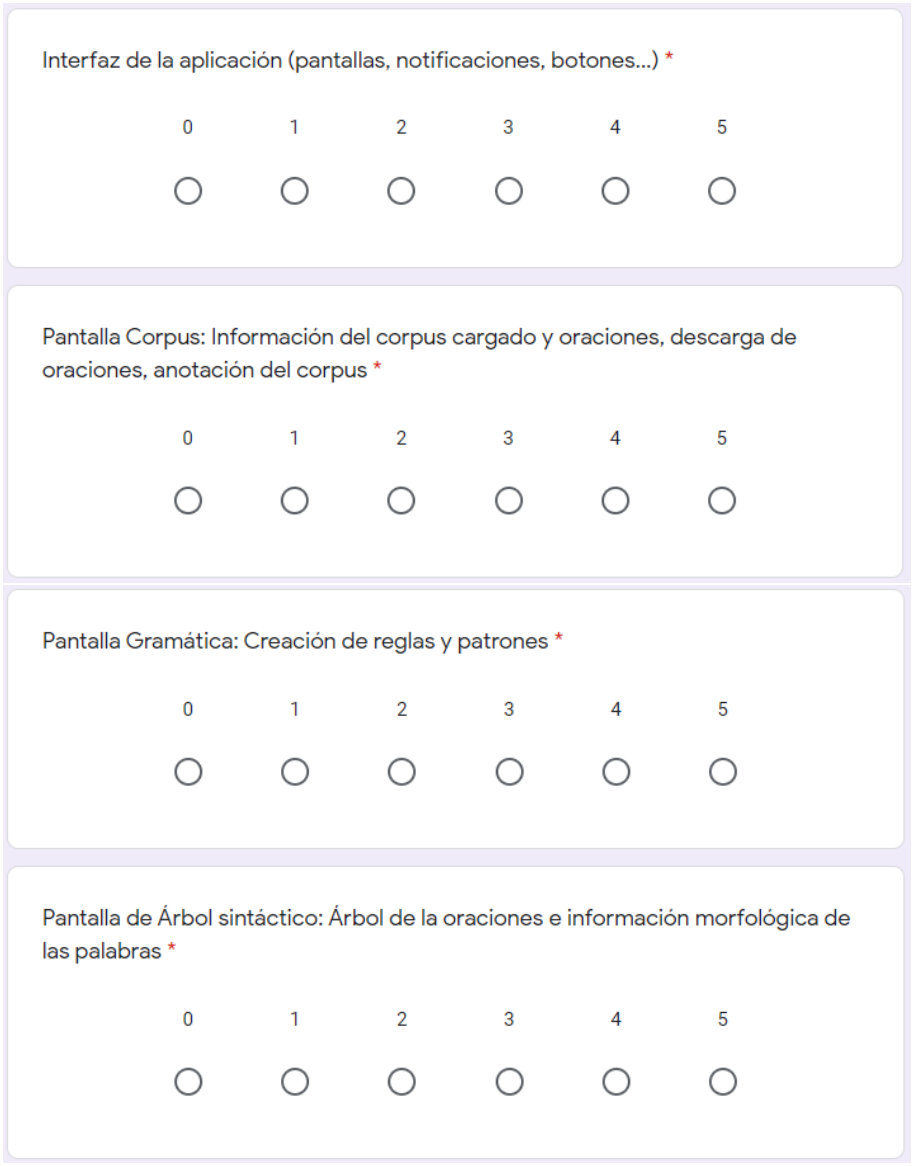


Figura 96: Preguntas para la evaluación de usuarios en el formulario 2

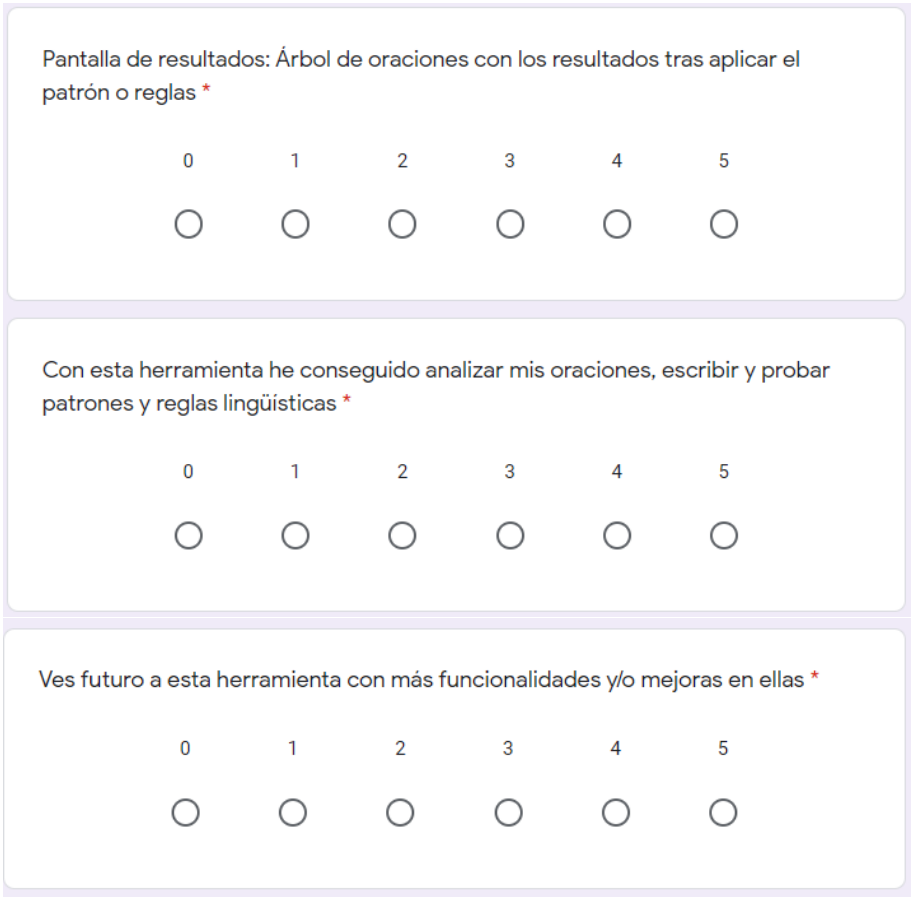


Figura 97: Preguntas para la evaluación de usuarios en el formulario 2